

# enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-06-12\*

©2024 by Pablo González†

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>5</b>	<b>The storage system</b>	<b>10</b>
1.1	Description and usage	2	5.1	Keys for storage system	10
1.2	The concept of left margin	3	5.1.1	Keys for label and ref	11
1.3	User interface	3	5.1.2	Keys for wrap and display	11
1.3.1	Internal counters	3	5.1.3	Keys for debug and checking	11
1.3.2	Support for multicol	3	5.2	The command <code>\anskey</code>	12
1.3.3	Support for minipage	3	5.2.1	Keys for <code>\anskey</code>	12
1.3.4	The <code>\label</code> and <code>\ref</code> system	4	5.3	The environment <code>anskey*</code>	13
1.3.5	Support for <code>\footnote</code>	4	5.4	The environment <code>keyans</code>	13
<b>2</b>	<b>The environments provided</b>	<b>4</b>	5.4.1	The <code>\item*</code> in <code>keyans</code>	14
2.1	The environment <code>enumext</code>	4	5.5	The environment <code>keyanspic</code>	14
2.2	The environment <code>enumext*</code>	5	5.5.1	The command <code>\anspic</code>	15
2.3	The command <code>\item*</code>	5	5.6	Printing stored content	15
2.3.1	Keys for <code>\item*</code>	5	5.6.1	The command <code>\getkeyans</code>	15
2.4	The command <code>\item</code> in <code>enumext*</code>	5	5.6.2	The command <code>\printkeyans</code>	16
<b>3</b>	<b>The command <code>\setenumext</code></b>	<b>6</b>	<b>6</b>	<b>Full examples</b>	<b>17</b>
<b>4</b>	<b>The keyval system</b>	<b>6</b>	<b>7</b>	<b>The way of non-enumerated lists</b>	<b>19</b>
4.1	Keys for label and ref	6	<b>8</b>	<b>References</b>	<b>21</b>
4.2	Keys for spaces	7	<b>9</b>	<b>Change history</b>	<b>22</b>
4.2.1	Vertical spaces	7	<b>10</b>	<b>Index of Documentation</b>	<b>23</b>
4.2.2	Horizontal spaces	8	<b>11</b>	<b>Implementation</b>	<b>25</b>
4.3	Keys for add code	8	<b>12</b>	<b>Index of Implementation</b>	<b>124</b>
4.4	Keys for start, series and resume	9			
4.5	Keys for multicol	9			
4.6	Keys for minipage	9			
4.6.1	The command <code>\miniright</code>	10			
4.6.2	The key <code>mini-right</code>	10			

## Motivation and acknowledgments

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

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

This package would not be possible without Phelype Oleinik who has collaborated and adapted a large part of the code and all  $\text{\LaTeX}$  team for their great work and to the different members of the `TeX-SX` community who have provided great answers and ideas. Here a note of the main ones:

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

\*This file describes a documentation for v1.0, last revised 2024-06-12.

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

1 Introduction

In the L<sup>A</sup>T<sub>E</sub>X world there are many useful packages and classes for creating “lists of exercises”, “worksheets” or “multiple choice questions”, classes like exam[1] and packages like xsim[2] do the job perfectly, but they don’t always fit the basic day to day needs.  
In my work (and in the work of many teachers) it is common to use “simple exercise sheets” also known as “informal lists of exercises”, as an example:

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

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

3. True False

(a)  $\alpha > \delta$

(b) L<sup>A</sup>T<sub>E</sub>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

B

A

A) B) C)

A

yellow duck

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 / 137

1. B),  $x = 5$

2. D)

3. C), some note
- \* 4. E), A duck

\* 5. D), “other note”

\*
- \*

\*

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

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

1.1 Description and usage

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

- This package is NOT intend to replace the `enumerate` environment nor replace the powerful `enumitem`[6], the approach is intended to work without hindering either of them.
- This package can be used with `xelatex`, `lualatex`, `pdflatex` and the classical `latex>dvips>ps2pdf` and is present in  $\text{\TeX}$  Live and  $\text{\MiKTeX}$ , use the package manager to install. For manual installation, download `enumext.zip` and unzip it, run `lualatex enumext.dtx` and move all files to appropriate locations, then run `mktexlsr`. To produce the documentation run `lualatex enumext.dtx` two times.

enumext.sty

enumext.pdf

README.md

enumext.dtx

>>

>>

>>

>>

TDS:tex/latex/enumext/

TDS:doc/latex/enumext/

TDS:doc/latex/enumext/

TDS:source/latex/enumext/

The package is loaded in the usual way:

```
\usepackage{enumext}
```

1.2 The concept of left margin

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

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



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

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



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

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



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

### 1.3 User interface

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

#### 1.3.1 Internal counters

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

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

#### 1.3.2 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.

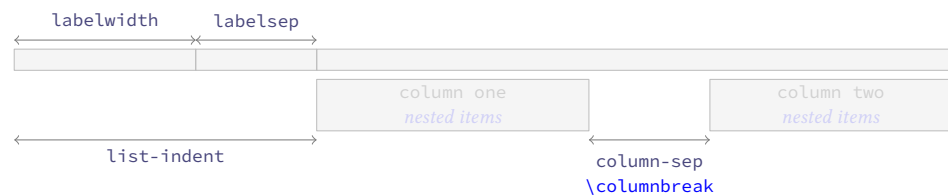


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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 §4.5).

- 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.

#### 1.3.3 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 §4.6).

#### 1.3.4 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 (§4.1), 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` (§5.1.1) when the key `save-ans` (§5.1) is active.

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

#### 1.3.5 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  $\text{\LaTeX}$ , `\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 second level.

i. This text is in the third level.
- A. This text is in the fourth level.

X This text is in the first 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, but it can be nested within `enumext` and can contain it nested within it.
- Each “item” in the environment is placed within a `minipage` environment whose *width* is stored in the dimension `\itemwidth` that includes `labelwidth`, `labelsep` plus 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.

X This text is in the first level.
2. 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 value set by the `labelsep` key and can be `\langle offset \rangle` using 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*={\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
  \item[Z] The ninth
  \item The tenth
\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               | The eighth item is way too long for this and needs Z | The ninth     |
|  | two columns     |  |               |
| 8. The tenth   |                 |  |               |

## 3 The command `\setenumext`

<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 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.



## 4.1 Keys for label and ref

`label = {⟨\alph* | \Alph* | \arabic* | \roman* | \Roman*⟩}` default: *by levels*

Sets the `⟨label⟩` 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 `(\alph*)`, 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 `⟨label⟩` 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 `⟨label⟩` as an argument, for example `\emph{⟨\alph*⟩}` will return an error. For full customization of how `⟨label⟩` is displayed use the `font` or `wrap-label` keys.

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

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

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

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

`labelsep = {⟨rigid length⟩}` default: `0.3333em`

Sets the *horizontal space* between the box containing the current `⟨label⟩` 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 = {⟨rigid length⟩}` default: *by label*

Sets the *width* of the box containing the current `⟨label⟩` 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 = {⟨integer | string⟩}` default: *empty*

Sets the `labelwidth` key pass the `⟨integer⟩` or converting the `⟨string⟩` 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 = {⟨font commands⟩}` default: *empty*

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

`align = {⟨left | right | center⟩}` default: *left*

Sets the *aligned* of `⟨label⟩` defined by `label` key on the current level in the label box.

`wrap-label = {⟨code {#1} more code⟩}` default: *empty*

Wraps the *current* `⟨label⟩` defined by `label` key referenced by `{#1}`. The `⟨code⟩` 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 \itembx { 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={\itembx{#1}}` or `wrap-label={\itembx*{#1}}`.

`wrap-label* = {⟨code {#1} more code⟩}` default: *empty*

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

## 4.2 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`.

### 4.2.1 Vertical spaces

`topsep` = { $\langle$ rubber length | rigid length $\rangle$ } 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` = { $\langle$ rubber length | rigid length $\rangle$ } 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` = { $\langle$ rubber length | rigid length $\rangle$ } 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 `keyans`, `keyans*` and `enumext*` environments the default value is 2.0pt plus 1.0pt minus 1.0pt.

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

`itemsep` = { $\langle$ rubber length | rigid length $\rangle$ } 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`  $\langle$ value forbidden $\rangle$  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`  $\langle$ value forbidden $\rangle$  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`  $\langle$ value forbidden $\rangle$  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  $\langle$ keys $\rangle$  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 ‘\*’  $\langle$ keys $\rangle$  applies `\vspace*` so that T<sub>E</sub>X does *not discard* this space at page break.

`above` = { $\langle$ rubber length | rigid length $\rangle$ } default: *not used*

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

`above*` = { $\langle$ rubber length | rigid length $\rangle$ } default: *not used*

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

`below` = { $\langle$ rubber length | rigid length $\rangle$ } default: *not used*

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

`below*` = { $\langle$ rubber length | rigid length $\rangle$ } default: *not used*

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



### 4.2.2 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.

### 4.3 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 `\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:  $\{\langle code \rangle\}\list{\langle arg one \rangle}{\langle arg two \rangle}$ .
- `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: \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.

### 4.4 Keys for start, series and resume

- `start` = { $\langle integer \mid string \rangle$ } default: 1  
 Sets the *start value* of the numbering on the current level. Internally  $\langle string \rangle$  is 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  $\langle keys \rangle$  are “*only*” available for the “*first level*” of `enumext` and `enumext*` and are ignored if set when nested inside each other.
- `series` = { $\langle series\ name \rangle$ } default: not used  
 Stores the *keys* of the optional argument of the “*first level*” of the environment in which it is executed in  $\{\langle series\ name \rangle\}$  which is used as an argument in the key `resume`. The  $\langle keys \rangle$  stored in  $\{\langle series\ name \rangle\}$  are not cumulative and are overwritten if the same  $\{\langle series\ name \rangle\}$  is used again.
- `resume` = { $\langle series\ name \rangle$ } default: not used  
 Sets the *start value* and *options* for the “*first level*” continuing the numbering of the environment in which the `series=\{\langle series\ name \rangle\}` key was executed. If passed *without value* this will only set *start value* continue the numbering from the last environment in which `series=\{\langle series\ name \rangle\}` or `resume=\{\langle series\ name \rangle\}` 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 the `start` key.

`resume*`  $\langle \text{value forbidden} \rangle$  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 the `start` key.

- For security reasons the `series` key will never save in  $\langle \text{series name} \rangle$  the keys `series`, `resume`, `resume*`, `save-ans`, `save-key` and `start`. When using the key `resume={\series name}` it will have hierarchy in the  $\langle \text{keys} \rangle$  that are saved in  $\langle \text{series name} \rangle$ , in order to establish the value of a  $\langle \text{key} \rangle$  already saved in  $\langle \text{series name} \rangle$  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*.

## 4.5 Keys for multicol

`columns` =  $\langle \text{integer} \rangle$  default: **1**

Set the *number of columns* to be used by the `multicol` environment within the environment. The value must be a positive integer less than or equal to **10**.

`columns-sep` =  $\langle \text{rigid length} \rangle$  default: *by level*

Set the *space between columns* used by the `multicol` 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 `multicol` 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.

## 4.6 Keys for minipage

`mini-env` =  $\langle \text{rigid length} \rangle$  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` =  $\langle \text{rigid length} \rangle$  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`.

### 4.6.1 The command \miniright

---

```
\miniright \begin{enumext}[mini-env=\langle rigid length \rangle] \item's before \item \miniright \langle content \rangle \end{enumext}
\begin{enumext}[mini-env=\langle rigid length \rangle] \item's before \item \miniright* \langle content \rangle \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 §1.3.5 for full support).

### 4.6.2 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` =  $\langle \text{content} \rangle$  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  $\langle \text{content} \rangle$  must be passed between braces.

`mini-right*` =  $\langle \text{code for drawing or tabular} \rangle$  default: *not used*

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

- The keys `mini-right` and `mini-right*` has a *slightly different* implementation, the argument  $\langle \text{content} \rangle$  is saved in a box and then printed outside the environment using *hooks*.

## 5 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={\langle store name \rangle}]
  \item Text \anskey{answer}
  \item Text
    \begin{keyans}
      ...
    \end{keyans}
\end{enumext}
```

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

By executing the key `save-ans={\langle store name \rangle}` 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  $\langle sequence \rangle$ .

### 5.1 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 = {\langle store name \rangle}` 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 = {\langle key list \rangle}` 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={\langle store name \rangle}]
  \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 = {\langle text symbol \rangle}` 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.

#### 5.1.1 Keys for label and ref

`save-ref = {\langle true | false \rangle}` 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{\langle store name : position \rangle}`, 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 = {\langle symbol \rangle}` 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.

### 5.1.2 Keys for wrap and display

- `wrap-ans` = {`<code> {#1} more code`} default: `\fbox{#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.

### 5.1.3 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>`}.

## 5.2 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.

### 5.2.1 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}
```

- |  |   |
|--|---|
| <ul style="list-style-type: none"> <li>★ 1. Text containing our instructions or questions.</li> <li>★ <span style="border: 1px solid black; padding: 2px;">first answer</span></li> <li>2. Text containing our instructions or questions.</li> <li>    (a) Question.</li> <li>    ★ <span style="border: 1px solid black; padding: 2px;">second answer</span></li> </ul> | <ul style="list-style-type: none"> <li>3. Text containing our instructions or questions.</li> <li>★ <span style="border: 1px solid black; padding: 2px;">third answer</span></li> <li>4. Text containing our instructions or questions.</li> <li>★ <span style="border: 1px solid black; padding: 2px;">fourth answer</span></li> </ul> |
|--|---|

## 5.3 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.

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

- 🔒 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*}
  \item Text containing our instructions or questions.
    \begin{enumext}
      \item Question.
        \begin{anskey*}
          \second answer
        \end{anskey*}
      \end{enumext}
  \item Text containing our instructions or questions.
    \begin{anskey*}
      \third answer
    \end{anskey*}
  \item Text containing our instructions or questions.
```



```

\begin{anskey*}
  \langle fourth answer \rangle
\end{anskey*}
\end{enumext}

```

- |   |  |
|---|--|
| <p>★ 5. Text containing our instructions or questions.</p> <p>[5] <span style="border: 1px solid black; padding: 2px;">First answer with verbatim</span></p> <p>6. Text containing our instructions or questions.</p> <p>    (a) Question.</p> <p>    [6] <span style="border: 1px solid black; padding: 2px;">second answer</span></p> | <p>7. Text containing our instructions or questions.</p> <p>[7] <span style="border: 1px solid black; padding: 2px;">third answer</span></p> <p>8. Text containing our instructions or questions.</p> <p>[8] <span style="border: 1px solid black; padding: 2px;">fourth answer</span></p> |
|---|--|

## 5.4 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 content \rangle]` is available for the `keyans*` environment.

<pre> \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} </pre>	<pre> \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} </pre>
--	--

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*`.

### 5.4.1 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
    \item Choice
  \end{keyans*}
  \item Text containing a question and image.
  \begin{keyans}[nosep,mini-env={0.4\linewidth}]
    \item Choice
    \item Choice
    \item Choice
    \item Choice
    \item*[\langle note \rangle] Correct choice
    \miniright
    \includegraphics[scale=0.25]{example-image-a}
    Some text
  \end{keyans}
\end{enumext}

```



```
\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

5.5 The environment keyanspic

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

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

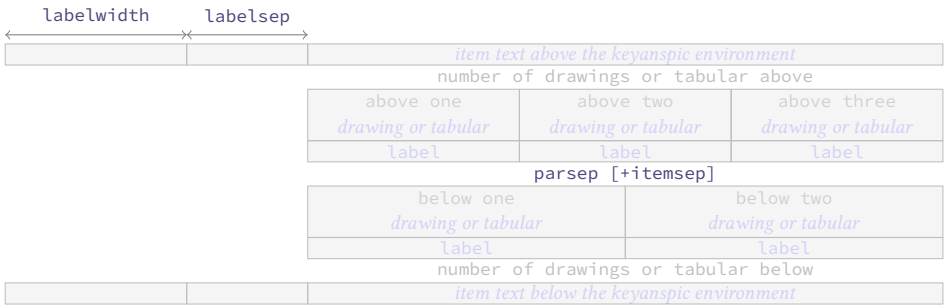


Figure 6: Representation of the `keyanspic` environment with optional argument [3,2] in `enumext`.

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

5.5.1 The command \anspic

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

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

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

Example

```
\begin{enumext}[save-ans=test,show-ans,nosep]  
  \item Question with images.  
    \begin{keyanspic}[3,2]  
      \anspic{\includegraphics[scale=0.15]{example-image-a}}  
      \anspic{\includegraphics[scale=0.15]{example-image-b}}  
      \anspic{\includegraphics[scale=0.15]{example-image-a}}  
      \anspic{\includegraphics[scale=0.15]{example-image-a}}  
      \anspic*[note]{\includegraphics[scale=0.15]{example-image-a}}  
    \end{keyanspic}  
  \end{enumext}
```

1. Question with images.

  
A)

  
B)

  
C)

  
D)

  
\* E)[note]

## 5.6 Printing stored content

### 5.6.1 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.

### 5.6.2 The command `\printkeyans`

---

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

---

The command `\printkeyans` prints “all stored content” in *sequence* `{<store name>}` defined by `save-ans` key placing this inside the `enumext` environment or the `enumext*` environment if the *starred argument* ‘\*’ is used. The “stored content” can only be accessed *after* it is stored in the *sequence*, if `{<store name>}` does not exist the command will return an error.

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

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

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

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

The default values for the “first level” of `\printkeyans` commands and `\printkeyans*` are established using `\setenumext[<print>,<1>]{<keys>}` and `\setenumext[<print*>]{<keys>}`. If we need to set the `<keys>` for the environment `enumext` “saved” in the *sequence* `{<store name>}` we will use `\setenumext[<print>,<level>]{<keys>}` and if we need to set the `<keys>` for the environment `enumext*` “saved” in the *sequence* `{<store name>}` we will use `\setenumext[<print>,<*>]{<keys>}`.

### Example

```
\begin{enumext}[save-ans=sample,columns=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 \LaTeX2e\ 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}
```

```
\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
- \*

\*

\*

\*

\*

6 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:

```
$ pdfdetach -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:

36 km/h.

360 km/h.

27,8 km/h.

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

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

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

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

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

36 km/h.

360 km/h.

27,8 km/h.

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

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

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

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

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

2. A

3. B

4. A

Example 2

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

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

36 km/h.

360 km/h.

27,8 km/h.

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

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

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

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

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

- D

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

A

36 km/h.

☒ B

360 km/h.

C

27,8 km/h.

D

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

☒ A

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

B

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

C

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

D

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

2. A

3. B


4. A
- \*

\*

\*

\*

Example 3

A “simple multiple choice” test .

1. First type of questions

A

value

B

correct

C

value

D

value
2. Second type of questions

I.

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

II.

$\alpha = \delta$

III.

$\angle EDF = 45^\circ$

A

I only

B

II only

C

I and II only

D

I and III only

E

I, II, and III
3. Third type of questions

(1)

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

(2)

$\angle EDF = 45^\circ$

A

value

B

value

C

value

D

value

E

value
4. Question with image and label below:

A

A

D

B

B

A

C

E

B

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)  $\mathbb{E}\mathbb{T}\mathbb{E}\mathbb{X}$ ze is cool?



Related to Linux

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

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

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

Example 5

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

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.	3	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	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.	4	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.

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

7 The way of non-enumerated lists

It is possible to use (or abuse) the `enumext` environment to mimic *non-enumerated* list environments such as `itemize` and `description`, clearly the *keys* to “store answers”, the `keyans` and `keyanspic` environments lose their sense and it is not the focus of the main of this package, but, why not to do it? Here I leave as an example other uses of the `enumext` environment that can be helpful for specific purposes. The “trick” to generate these *fake environments* is set `label={}` or `label={\some}` and play with the `list-indent`, `list-offset`, `font` and `wrap-label` keys.

Fake itemize environment

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

- First level item
    - Second level item
      - \* Third level item
        - Fourth level item
  - First level item
- \* First level item
    - ◇ Second level item
      - Third level item
        - ★ Fourth level item
  - \* First level item

Fake description environment

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

**SomeThing** A short one-line description.  
This is an entry *without* a label.  
**Something** A short *one-line* description text.  
**Something long** A much *longer* description text may take more than one line or more than one paragraph.  
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

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

**SomeThing** A short one-line description.  
This is an entry *without* a label.  
**Something** A short *one-line* description text.  
**Something long** A much *longer* description text may take more than one line or more than one paragraph.  
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

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

Description indented by label

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

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

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

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

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

**SomeThing** A short one-line description.  
This is an entry *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 \itembx { 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*={\itembx{#1}}`.

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

This is an entry *without* a label.

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

**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** 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** vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.

## Final notes

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

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

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

### Why list environments?

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

Of course, seeing the final result of the experiment “*reinventing the wheel*” I am quite satisfied.

### Why not random questions and other utilities

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

## 8 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<sub>ε</sub> 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.

## 9 Change history

**v1.0 2024-06-12** – First public release.

10 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
\getkeyans	12, 16
\item*	5–7, 11, 12, 14
\itemwidth	5
\item	5–7, 9, 10, 12, 14
\miniright	10
\printkeyans	6, 11, 16
\setenumext	5–7, 11, 12, 14, 16
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
enumext	4–9, 11–14, 16, 19
keyans*	4–14
keyanspic	4, 6, 8, 11–13, 15, 19
keyans	4–9, 11–15, 19
Environments:	
enumerate	1, 3, 5, 21
figure	5
list	3, 9, 21
minipage	3–5, 10, 21
multicols	3, 4, 10
table	5
task	5
F	
\footnote	5
I	
\itemsep	8
K	
Keys for command provide by enumext:	
break-col	12
item-join	12
item-pos*	13
item-star	12, 13
item-sym*	13
Keys for environments provide by enumext:	
above*	8
above	8
after	9, 10
align	7, 20
base-fix	8
before*	9
before	9
below*	8
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, 5–7, 9, 10, 12, 20
labelwidth	3, 6, 7, 9, 10, 12, 20
labelwith	5
label	7, 9, 14, 19, 20
list-indent	3, 9
list-offset	3, 9, 20
listparindent	9
mark-ans	12
mark-pos	12
mark-ref	11
mini-env	4, 8, 10
mini-right*	6, 10
mini-right	6, 10
mini-sep	4, 10
no-store	11–13
noitemsep	8
nosep	8, 19
parsep	8, 15
partopsep	8
ref	4, 7
resume*	6, 10, 11
resume	6, 9–11
rightmargin	9
save-ans	4, 6, 9–16
save-key	10, 11, 16
save-ref	4, 7, 11–13, 16
save-sep	11
series	6, 9–11
show-ans	11, 12
show-length	7
show-pos	11, 12, 16
start	9, 10
topsep	8
widest	7
wrap-ans	12
wrap-label*	7, 20
wrap-label	7
wrap-opt	12
L	
\label	4
Labels provide by enumext:	
\Alph*	7, 14
\Roman*	7
\alph*	7

<code>\arabic*</code> .....	7	<code>l3seq</code> .....	1, 21
<code>\roman*</code> .....	7	<code>multicol</code> .....	1, 2, 4, 21
<code>\labelsep</code> .....	3, 7	<code>scontents</code> .....	1, 2, 13
<code>\labelwidth</code> .....	3, 7	<code>task</code> .....	5, 6
<code>\linewidth</code> .....	10	<code>xsim</code> .....	2
<code>\listparindent</code> .....	9	<code>\parsep</code> .....	8
<b>P</b>		<code>\partopsep</code> .....	8
Packages:			
<code>enumerate</code> .....	21	<b>R</b>	
<code>enumext</code> .....	1–6, 15, 21	<code>\raggedcolumns</code> .....	4
<code>enumitem</code> .....	3–5, 9, 20, 21	<code>\ref</code> .....	4
<code>footnotehyper</code> .....	4, 5	<code>\rightmargin</code> .....	9
<code>hyperref</code> .....	4, 5, 11–13, 21	<b>T</b>	
<code>l3keys</code> .....	6	<code>\topsep</code> .....	8
<code>l3prop</code> .....	1, 21		

## 11 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  $\text{\TeX}$ pert are superfluous, but, after so many years developing this project is the only way to remember what does what.

### 11.1 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.

### 11.2 Initial set up

Start the DocStrip guards.

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

Identify the internal prefix ( $\text{\LaTeX}$ 3 DocStrip convention) for `l3doc` class.

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

### 11.3 Declaration of the package

First we will make sure we have a minimum (super updated) version of  $\text{\LaTeX}$  to work correctly.

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

Now declare the `enumext` package.

```
4 \ProvidesExplPackage
5   {enumext}
6   {2024-06-12}
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 }
```

## 11.4 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
```

Internal variables used by functions `\__enumext_is_not_nested:`, `\__enumext_is_on_first_level:` and `\__enumext_keyans_start_line:` (§11.5.1).

```
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
```

(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` (§11.9) and then modified by the function `\__enumext_label_style:Nnn` used by `label` key (§11.12).

```
44 \cs_set_protected:Npn \__enumext_tmp:n #1
45 {
46   \tl_new:c { l__enumext_counter_#1_tl }
47 }
48 \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 (§11.12).

```
49 \tl_const:Nn \c__enumext_counter_style_tl
50 { { arabic } { roman } { Roman } { alph } { Alph } }
51 \tl_new:N \l__enumext_ref_key_arg_tl
52 \tl_new:N \l__enumext_ref_the_count_tl
53 \cs_set_protected:Npn \__enumext_tmp:n #1
54 {
55   \tl_new:c { l__enumext_renew_the_count_#1_tl }
56   \tl_new:c { l__enumext_the_counter_#1_tl }
57   \tl_set:ce { l__enumext_the_counter_#1_tl } { \exp_not:c { theenumX#1 } }
58 }
59 \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
\g__enumext_item_symbol_tl
```

Internal variables used by `resume`, `resume*` and `series` keys (§11.23).

```
60 \int_new:N \g__enumext_resume_int
61 \int_new:N \g__enumext_resume_vii_int
62 \tl_new:N \l__enumext_resume_name_tl
63 \bool_new:N \l__enumext_resume_active_bool
64 \tl_new:N \g__enumext_standar_series_tl
65 \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` (§11.13) and `label` (§11.11) keys.

```

66 \dim_new:N \l__enumext_current_widest_dim
67 \tl_new:N \g__enumext_counter_styles_tl
68 \tl_new:N \g__enumext_widest_label_tl
69 \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 (§11.16). The variables `\l__enumext_leftmargin_X_dim` and `\l__enumext_itemindent_X_dim` are used and set by the function `\__enumext_calc_hspace:NNNNNNNNNN` (§11.34.1).

```

70 \cs_set_protected:Npn \__enumext_tmp:n #1
71 {
72   \bool_new:c { \l__enumext_leftmargin_tmp_#1_bool }
73   \dim_new:c { \l__enumext_leftmargin_tmp_#1_dim }
74   \dim_new:c { \l__enumext_leftmargin_#1_dim }
75   \dim_new:c { \l__enumext_itemindent_#1_dim }
76 }
77 \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

```

Internal variables used by `columns` key §11.20).

```

78 \cs_set_protected:Npn \__enumext_tmp:n #1
79 {
80   \skip_new:c { \l__enumext_multicols_above_#1_skip }
81   \skip_new:c { \l__enumext_multicols_below_#1_skip }
82 }
83 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

(End of definition for `\l__enumext_multicols_above_X_skip` and `\l__enumext_multicols_below_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 (§11.21.4) and the keys `mini-right`, `mini-right*`, `mini-env` and `mini-sep` (§11.19, §11.21).

```

84 \int_new:N \g__enumext_minipage_stat_int
85 \skip_new:N \l__enumext_minipage_left_skip
86 \skip_new:N \l__enumext_minipage_right_skip
87 \skip_new:N \l__enumext_minipage_after_skip
88 \skip_new:N \g__enumext_minipage_right_skip
89 \skip_new:N \g__enumext_minipage_after_skip
90 \cs_set_protected:Npn \__enumext_tmp:n #1
91 {
92   \dim_new:c { \l__enumext_minipage_left_#1_dim }
93   \bool_new:c { \l__enumext_minipage_active_#1_bool }
94 }
95 \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 integer variable `\l__enumext_start_X_int` are used by the `start` key (§11.13), the token list `\l__enumext_fake_item_indent_X_tl` is used by `itemindent` key (§11.16.1), the variables `\l__enumext_label_fill_left_X_tl` and `\l__enumext_label_fill_right_X_tl` are used by the `align` key (§11.11). 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 (§11.18).

```

96 \cs_set_protected:Npn \__enumext_tmp:n #1
97 {
98   \bool_new:c { \l__enumext_wrap_label_#1_bool }
99   \bool_new:c { \l__enumext_wrap_label_opt_#1_bool }
100   \int_new:c { \l__enumext_start_#1_int }
101   \tl_new:c { \l__enumext_fake_item_indent_#1_tl }
102   \tl_new:c { \l__enumext_label_fill_left_#1_tl }
103   \tl_new:c { \l__enumext_label_fill_right_#1_tl }
104   \bool_new:c { \l__enumext_vspace_a_star_#1_bool }
105   \bool_new:c { \l__enumext_vspace_b_star_#1_bool }
106 }
107 \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 (§11.24.1) 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` (§11.27) 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*` (§11.28).

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*` (§11.32.2) and `\anspic*` (§11.37.1) 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.

```
108 \bool_new:N \l__enumext_store_active_bool
109 \tl_new:N \l__enumext_store_name_tl
110 \tl_new:N \g__enumext_store_name_tl
111 \tl_new:N \l__enumext_store_anskey_arg_tl
112 \tl_new:N \l__enumext_store_anskey_env_tl
113 \tl_new:N \l__enumext_store_anskey_opt_tl
114 \tl_new:N \l__enumext_store_current_label_tl
115 \tl_new:N \l__enumext_store_current_opt_arg_tl
116 \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` (§11.43).

```
117 \tl_new:N \l__enumext_setkey_tmpa_tl
118 \tl_new:N \l__enumext_setkey_tmpb_tl
119 \int_new:N \l__enumext_setkey_tmpa_int
120 \seq_new:N \l__enumext_setkey_tmpa_seq
121 \seq_new:N \l__enumext_setkey_tmpb_seq
```

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

```
\l__enumext_print_keyans_starred_tl
\l__enumext_mark_position_str
\g__enumext_item_symbol_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` (§11.42), `show-pos` key (§11.26), `item-sym*` key (§11.30), `save-key` key (§11.26.2) and “*storage level system*”.

```
122 \tl_new:N \l__enumext_print_keyans_starred_tl
123 \str_new:N \l__enumext_mark_position_str
124 \tl_new:N \g__enumext_item_symbol_tl
125 \cs_set_protected:Npn \__enumext_tmp:n #1
126 {
127   \tl_new:c { \l__enumext_print_keyans_#1_tl }
128   \tl_new:c { \l__enumext_store_save_key_#1_tl }
129   \bool_new:c { \l__enumext_store_save_key_#1_bool }
130   \bool_new:c { \l__enumext_store_upper_level_#1_bool }
131 }
132 \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 (§11.37.2).

```
133 \seq_new:N \l__enumext_keyans_pic_body_seq
134 \dim_new:N \l__enumext_keyans_pic_width_dim
135 \int_new:N \l__enumext_keyans_pic_above_int
136 \int_new:N \l__enumext_keyans_pic_below_int
137 \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 (§11.24.3) 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.

```
138 \bool_new:N \l__enumext_check_answers_bool
139 \bool_new:N \g__enumext_check_ans_key_bool
140 \tl_new:N \l__enumext_check_start_line_env_tl
141 \int_new:N \g__enumext_check_starred_cmd_int
142 \int_new:N \g__enumext_item_anskey_int
```

```

143 \int_new:N \g__enumext_item_number_int
144 \bool_new:N \l__enumext_item_number_bool
145 \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 (§11.8). The boolean variable `\l__enumext_footnotes_key_bool` determine if `hyperref` is load with key `hyperfootnotes=true`.

```

146 \bool_new:N \l__enumext_hyperref_bool
147 \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 (§11.26). 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` (§11.8) and the variable `\l__enumext_write_aux_file_tl` will be in charge of executing the writing code in the `.aux` file.

```

148 \tl_new:N \l__enumext_newlabel_arg_one_tl
149 \tl_new:N \l__enumext_newlabel_arg_two_tl
150 \tl_new:N \l__enumext_write_aux_file_tl
151 \cs_set_protected:Npn \__enumext_tmp:n #1
152 {
153   \tl_new:c { \l__enumext_label_copy_#1_tl }
154 }
155 \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` (§11.31).

```

156 \int_new:N \g__enumext_footnote_int
157 \seq_new:N \g__enumext_footnote_arg_seq
158 \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_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.

```

159 \cs_set_protected:Npn \__enumext_tmp:n #1
160 {
161   \bool_new:c { \l__enumext_item_starred_#1_bool }
162   \int_new:c { \l__enumext_item_column_pos_#1_int }
163   \int_new:c { \g__enumext_item_count_all_#1_int }
164   \int_new:c { \l__enumext_joined_item_#1_int }
165   \int_new:c { \l__enumext_joined_item_aux_#1_int }
166   \int_new:c { \l__enumext_tmpa_#1_int }
167   \box_new:c { \l__enumext_item_text_#1_box }
168   \dim_new:c { \l__enumext_joined_width_#1_dim }
169   \dim_new:c { \l__enumext_item_width_#1_dim }
170   \tl_new:c { \g__enumext_item_symbol_aux_#1_tl }
171   \str_new:c { \l__enumext_align_label_#1_str }
172   \bool_new:c { \g__enumext_minipage_active_#1_bool }
173   \box_new:c { \l__enumext_miniright_code_#1_box }
174   \bool_new:c { \g__enumext_minipage_center_#1_bool }
175   \dim_new:c { \g__enumext_minipage_right_#1_dim }
176   \skip_new:c { \g__enumext_minipage_right_#1_skip }
177 }
178 \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`.

```

179 \clist_const:Nn \c__enumext_all_envs_clist
180 {
181   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
182   {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
183 }

```

(End of definition for `\c__enumext_all_envs_clist`.)

## 11.5 Some utility functions

`\__enumext_at_begin_document:n`

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

```
184 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
185 {
186   \hook_gput_code:nnn {begindocument} {enumext} { #1 }
187 }
```

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

`\__enumext_after_env:nn`  
`\__enumext_before_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.

```
188 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
189 {
190   \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
191 }
192 \cs_new_protected:Npn \__enumext_before_env:nn #1 #2
193 {
194   \hook_gput_code:nnn {env/#1/before} {enumext} {#2}
195 }
```

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

`\__enumext_level:`

Function for check current level in `enumext`.

```
196 \cs_new:Nn \__enumext_level:
197 {
198   \int_to_roman:n { \l__enumext_level_int }
199 }
```

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

`\__enumext_if_is_int:nT`  
`\__enumext_if_is_int:nF`  
`\__enumext_if_is_int:nTF`

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?](#).

```
200 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
201 {
202   \regex_match:nnTF { ^[\+|-]?[\d]+$ } {#1} % $
203   { \prg_return_true: }
204   { \prg_return_false: }
205 }
```

(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.

```
206 \cs_new_protected:Nn \__enumext_regex_counter_style:
207 {
208   \tl_map_inline:Nn \c__enumext_counter_style_tl
209   {
210     \regex_replace_once:nnN { \c{##1}\* }
211     { \c{##1}\cB{\u{\l__enumext_ref_the_count_tl}\cE} } \l__enumext_ref_key_arg_tl
212   }
213 }
```

(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.

```
214 \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
215 {
216   * ~ #2
217   \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
218   = ~ \use:c { #1_use:c } { \l__enumext_#2_#3_#1 } \\
219 }
```

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

### 11.5.1 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.

```

220 \cs_new_protected:Nn \__enumext_is_not_nested:
221 {
222   \str_case:en { \@currentenv }
223   {
224     {enumext}
225     {
226       \bool_lazy_and:nnT
227       { \bool_not_p:n { \g__enumext_standar_bool } }
228       { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
229       {
230         \bool_gset_true:N \g__enumext_standar_bool
231       }
232     }
233     {enumext*}
234     {
235       \bool_lazy_and:nnT
236       { \bool_not_p:n { \g__enumext_starred_bool } }
237       { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
238       {
239         \bool_gset_true:N \g__enumext_starred_bool
240       }
241     }
242   }
243 }

```

The function `\__enumext_is_on_first_level:` will set the variables `\l__enumext_standar_first_bool` (§11.24.1), `\l__enumext_starred_first_bool` (§11.24.1) and `\l__enumext_anskey_env_bool` (§11.28) 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.

```

244 \cs_new_protected:Nn \__enumext_is_on_first_level:
245 {
246   \bool_lazy_all:nT
247   {
248     { \bool_if_p:N \g__enumext_standar_bool }
249     { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
250     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
251   }
252   {
253     \bool_set_true:N \l__enumext_standar_first_bool
254     \bool_set_true:N \l__enumext_anskey_env_bool
255     \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
256     \tl_gset:Nn \g__enumext_start_line_tl
257     {
258       on ~ line ~ \exp_not:V \inputlineno
259     }
260   }
261   \bool_lazy_all:nT
262   {
263     { \bool_if_p:N \g__enumext_starred_bool }
264     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
265     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
266   }
267   {
268     \bool_set_true:N \l__enumext_starred_first_bool
269     \bool_set_true:N \l__enumext_anskey_env_bool
270     \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
271     \tl_gset:Nn \g__enumext_start_line_tl
272     {
273       on ~ line ~ \exp_not:V \inputlineno
274     }
275   }
276 }

```

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

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

```

277 \cs_new_protected:Nn \__enumext_keyans_start_line:
278 {
279   \str_case:en { \@currentenvir }
280   {
281     {keyans}
282     {
283       \tl_set:Nc \l__enumext_check_start_line_env_tl
284       {
285         in ~ 'keyans' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
286       }
287     }
288     {keyans*}
289     {
290       \tl_set:Nc \l__enumext_check_start_line_env_tl
291       {
292         in ~ 'keyans*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
293       }
294     }
295     {keyanspic}
296     {
297       \tl_set:Nc \l__enumext_check_start_line_env_tl
298       {
299         in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
300       }
301     }
302   }
303 }

```

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

### 11.5.2 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:
304 \cs_new_protected:Nn \__enumext_reset_global_vars:
305 {
306   \__enumext_reset_global_int:
307   \__enumext_reset_global_bool:
308   \__enumext_reset_global_tl:
309 }
310 \cs_new_protected:Nn \__enumext_reset_global_int:
311 {
312   \int_gzero:N \g__enumext_item_number_int
313   \int_gzero:N \g__enumext_item_anskey_int
314   \int_gzero:N \g__enumext_item_answer_diff_int
315 }
316 \cs_new_protected:Nn \__enumext_reset_global_bool:
317 {
318   \bool_gset_false:N \g__enumext_check_ans_key_bool
319   \bool_gset_false:N \g__enumext_standar_bool
320   \bool_gset_false:N \g__enumext_starred_bool
321 }
322 \cs_new_protected:Nn \__enumext_reset_global_tl:
323 {
324   \tl_gclear:N \g__enumext_store_name_tl
325   \tl_gclear:N \g__enumext_start_line_tl
326   \tl_gclear:N \g__enumext_envir_name_tl
327 }

```

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

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

```

328 \cs_new_protected:Nn \__enumext_log_global_vars:
329 {
330   \msg_log:nneeee { enumext } { prop-seq-int-hook }
331   { \g__enumext_store_name_tl }

```



```

332     { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
333     { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
334     { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
335 }

```

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.

```

336 \cs_new_protected:Nn \__enumext_log_answer_vars:
337 {
338     \msg_log:nneee { enumext } { item-answer-hook }
339     { \int_use:N \g__enumext_item_number_int }
340     { \int_use:N \g__enumext_item_anskey_int }
341     { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int } }
342 }

```

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

## 11.6 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.

```

343 \__enumext_at_begin_document:n
344 {
345     \cs_new_eq:NN \__enumext_start_list:nn \list
346     \cs_new_eq:NN \__enumext_stop_list: \endlist
347     \cs_new_eq:NN \__enumext_item_std:w \item
348 }

```

(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.

```

349 \__enumext_at_begin_document:n
350 {
351     \cs_new_eq:NN \__enumext_minipage:w \minipage
352     \cs_new_eq:NN \__enumext_endminipage: \endminipage
353 }

```

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

## 11.7 The internal minipage environment

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 `\vspace{0pt}` to maintain alignment on “top”. 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 (§11.35) and `\__enumext_safe_exec_vii:` in the `enumext*` environment definition (§11.39)

```

354 \cs_new_protected:Nn \__enumext_internal_mini_page:
355 {
356     \int_compare:nNtT { \l__enumext_level_int } = { 0 }
357     {
358         \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
359         {

```

```

360         \__enumext_minipage:w [ t ] { ##1 }
361         \legacy_if_gset_false:n { @minipage }
362         \vspace { 0pt }
363     }
364     { \__enumext_endminipage: }
365 }
366 }

```

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

## 11.8 Compatibility with hyperref and footnotehyper

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

```

367 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
368 \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_hyperref_bool` to “true” if the package is loaded. At this point we will use the public macro `\IfHyperBoolean` to determine if the `hyperfootnotes=true` key is present, if so, we set the state of the boolean variable `\__enumext_footnotes_key_bool` to “true”.

```

369 \cs_new_protected:Nn \__enumext_after_hyperref:
370 {
371     \IfPackageLoadedTF { hyperref }
372     {
373         \msg_info:nnn { enumext } { package-load } { hyperref }
374         \bool_set_true:N \l__enumext_hyperref_bool
375         \IfHyperBoolean{hyperfootnotes}
376         {
377             \typeout{hyperfootnotes=true}
378             \bool_set_true:N \l__enumext_footnotes_key_bool
379         }
380         { \typeout{hyperfootnotes=false} }
381     }
382     { }

```

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`.

```

383 \bool_if:NT \l__enumext_footnotes_key_bool
384 {
385     \IfPackageLoadedTF { footnotehyper }
386     {
387         \msg_info:nnn { enumext } { package-load } { footnotehyper }
388     }
389     {
390         \typeout{No ~ footnotehyper ~ load}
391         \typeout{Load ~ and ~ use ~ \string\makesavenoteenv{enumext*}}
392         \bool_set_false:N \l__enumext_footnotes_key_bool
393     }
394 }

```

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

```

395 \bool_if:NTF \l__enumext_hyperref_bool
396 {
397     \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
398     \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
399 }
400 {
401     \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
402     \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
403 }
404 }

```

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

```
\__enumext_newlabel:nn
```

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

**#1:** `\l__enumext_newlabel_arg_one_tl`

#2: `\l__enumext_newlabel_arg_two_tl`

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

```

405 \cs_new_protected:Npn \l__enumext_newlabel:nn #1 #2
406 {
407   \protected@write \@auxout { }
408   {
409     \token_to_str:N \newlabel {#1}
410     {
411       {#2}
412       \bool_if:NT \l__enumext_hyperref_bool
413       { { \thepage } {#2} {#1} }
414       { }
415     }
416   }
417   \__enumext_hypertarget:nn {#1} { }
418   \__enumext_phantomsection:
419 }

```

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

## 11.9 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 `\l__enumext_counter_X_tl` for “*store*” the counter’s name.

#2: The counter’s name.

```

420 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
421 {
422   \cs_if_exist:cTF { c@ #2 }
423   { \msg_fatal:nnn { enumext } { counters } { #2 } }
424   {
425     \tl_set:Nn #1 { #2 }
426     \newcounter { #2 }
427   }
428 }

```

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

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

```

enumXi   429 \__enumext_define_counters:Nn \l__enumext_counter_i_tl   { enumXi   }
enumXii  430 \__enumext_define_counters:Nn \l__enumext_counter_ii_tl  { enumXii  }
enumXiii 431 \__enumext_define_counters:Nn \l__enumext_counter_iii_tl { enumXiii }
enumXiv  432 \__enumext_define_counters:Nn \l__enumext_counter_iv_tl  { enumXiv  }
enumXvii 433 \__enumext_define_counters:Nn \l__enumext_counter_v_tl   { enumXv   }
enumXviii 434 \__enumext_define_counters:Nn \l__enumext_counter_vi_tl   { enumXvi  }
          435 \__enumext_define_counters:Nn \l__enumext_counter_vii_tl  { enumXvii }
          436 \__enumext_define_counters:Nn \l__enumext_counter_viii_tl { enumXviii }

```

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

## 11.10 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*`, `\Alpha*`, `\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.

```

437 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
438 {
439   \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
440   \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
441 }
442 \__enumext_register_counter_style:Nn \arabic { 0 }
443 \__enumext_register_counter_style:Nn \Alpha { M }
444 \__enumext_register_counter_style:Nn \alph { m }
445 \__enumext_register_counter_style:Nn \Roman { VIII }
446 \__enumext_register_counter_style:Nn \roman { viii }

```

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

`\__enumext_label_width_by_box:Nn`  
`\__enumext_label_width_by_box:cv`

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

```
447 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
448 {
449   \hbox_set:Nn \__enumext_label_width_by_box {#2}
450   \dim_set:Nn #1 { \box_wd:N \__enumext_label_width_by_box }
451 }
452 \cs_generate_variant:Nn \__enumext_label_width_by_box:Nn { cv }
```

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

`\__enumext_label_style:Nnn`  
`\__enumext_label_style:cvn`

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

```
453 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
454 {
455   \tl_clear_new:N #1
456   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
457   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
458   \tl_map_inline:Nn \g__enumext_counter_styles_tl
459   {
460     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
461     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
462     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
463   }
464   \__enumext_label_width_by_box:Nn \__enumext_current_widest_dim
465   { \tl_use:N \g__enumext_widest_label_tl }
466   \tl_set_eq:cN { the #2 } #1
467 }
468 \cs_generate_variant:Nn \__enumext_label_style:Nnn { cvn }
```

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

### 11.11 Setting keys associated with label

`font`  
`labelsep`  
`labelwidth`  
`wrap-label`  
`wrap-label*`

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

```
469 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
470 {
471   \keys_define:nn { enumext / #1 }
472   {
473     font      .tl_set:c   = { l__enumext_label_font_style_#2_tl },
474     font      .value_required:n = true,
475     labelsep  .dim_set:c   = { l__enumext_labelsep_#2_dim },
476     labelsep  .initial:n   = {0.3333em},
477     labelsep  .value_required:n = true,
478     labelwidth .dim_set:c   = { l__enumext_labelwidth_#2_dim },
479     labelwidth .value_required:n = true,
480     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
481     wrap-label .initial:n   = {##1},
482     wrap-label .value_required:n = true,
483     wrap-label* .code:n = {
484       \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
485       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
486     },
487     wrap-label* .value_required:n = true,
488   }
489 }
490 \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.

```

491 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
492 {
493   \keys_define:nn { enumext / #1 }
494   {
495     align .choice:,
496     align / left .code:n =
497       {
498         \tl_clear:c { l__enumext_label_fill_left_#2_tl }
499         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
500       },
501     align / right .code:n =
502       {
503         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
504         \tl_clear:c { l__enumext_label_fill_right_#2_tl }
505       },
506     align / center .code:n =
507       {
508         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
509         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
510       },
511     align / unknown .code:n =
512       \msg_error:nneee { enumext } { unknown-choice }
513       { align } { left, ~ right, ~ center } { \exp_not:n {##1} },
514     align .initial:n = left,
515     align .value_required:n = true,
516   }
517 }
518 \clist_map_inline:nn
519 {
520   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
521 }
522 { \__enumext_tmp:nn #1 }

523 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
524 {
525   \keys_define:nn { enumext / #1 }
526   {
527     align .choice:,
528     align / left .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
529     align / right .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
530     align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
531     align / unknown .code:n =
532       \msg_error:nneee { enumext } { unknown-choice }
533       { align } { left, ~ right, ~ center } { \exp_not:n {##1} },
534     align .initial:n = left,
535     align .value_required:n = true,
536   }
537 }
538 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

(End of definition for `align`.)

## 11.12 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.

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

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

```

\l__enumext_label_i_tl
\l__enumext_label_ii_tl
\l__enumext_label_iii_tl
\l__enumext_label_iv_tl

539 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
540 {
541   \keys_define:nn { enumext / #1 }
542   {
543     label .code:n = {
544       \__enumext_label_style:cnv { l__enumext_label_#2_tl }
545       { l__enumext_counter_#2_tl } {##1}
546       \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
547       \l__enumext_current_widest_dim

```

```

548         },
549         label .initial:n = #3,
550         label .value_required:n = true,
551         ref .code:n = \__enumext_standar_ref:n {##1},
552         ref .value_required:n = true,
553     }
554 }
555 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
556 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*. ) }
557 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
558 \__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }

```

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

```

\__enumext_standar_ref:n
\__enumext_standar_ref:

```

The `\__enumext_standar_ref:n` first we will pass the key argument to `\l__enumext_ref_key_arg_tl` and we will analyze its state, if it is not *empty* we will make a copy of the current counter in `\l__enumext_ref_the_count_tl` and we will execute the function `\__enumext_regex_counter_style:` which will return the modified `\l__enumext_ref_key_arg_tl` and we make the value of `\l__enumext_ref_the_count_tl` the same as that `\l__enumext_the_counter_X_tl` which contains `\theenumX` and finally we set `\l__enumext_renew_the_count_X_tl` with the renewed command.

```

559 \cs_new_protected:Npn \__enumext_standar_ref:n #1
560 {
561     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
562     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
563     {
564         \msg_error:nnn { enumext } { key-ref-empty } { enumext }
565     }
566     {
567         \tl_set_eq:Nc
568         \l__enumext_ref_the_count_tl { \l__enumext_counter_ \__enumext_level: _tl }
569         \__enumext_regex_counter_style:
570         \tl_set_eq:Nc
571         \l__enumext_ref_the_count_tl { \l__enumext_the_counter_ \__enumext_level: _tl }
572         \tl_put_right:ce { \l__enumext_renew_the_count_ \__enumext_level: _tl }
573         {
574             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
575             { \exp_not:V \l__enumext_ref_key_arg_tl }
576         }
577     }
578 }

```

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

```

579 \cs_new_protected:Npn \__enumext_standar_ref:
580 {
581     \tl_if_empty:cF { \l__enumext_renew_the_count_ \__enumext_level: _tl }
582     {
583         \tl_use:c { \l__enumext_renew_the_count_ \__enumext_level: _tl }
584     }
585 }

```

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

### 11.12.2 Define and set `label` and `ref` keys for `enumext*` and `keyans*` environments

`label` 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
586 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
587 {
588     \keys_define:nn { enumext / #1 }
589     {
590         label .code:n = {
591             \__enumext_label_style:cnv { \l__enumext_label_#2_tl }
592             { \l__enumext_counter_#2_tl } {##1}
593             \dim_set_eq:cN { \l__enumext_labelwidth_#2_dim }
594             \l__enumext_current_widest_dim
595         },
596         label .initial:n = #3,
597         label .value_required:n = true,
598         ref .code:n = \__enumext_starred_ref:n {##1},
599         ref .value_required:n = true,
600     }

```



```

601 }
602 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
603 \__enumext_tmp:nnn { keyans* } { viii } { \Alph*} }

```

(End of definition for label and others.)

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

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

```

604 \cs_new_protected:Npn \__enumext_starred_ref:n #1
605 {
606   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
607   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
608   {
609     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
610     {
611       \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
612     }
613     {
614       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
615       \__enumext_regex_counter_style:
616       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
617       \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
618       {
619         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
620         { \exp_not:V \l__enumext_ref_key_arg_tl }
621       }
622     }
623   }
624   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
625   {
626     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
627     {
628       \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
629     }
630     {
631       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
632       \__enumext_regex_counter_style:
633       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
634       \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
635       {
636         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
637         { \exp_not:V \l__enumext_ref_key_arg_tl }
638       }
639     }
640   }
641 }

```

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.

```

642 \cs_new_protected:Nn \__enumext_starred_ref:
643 {
644   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
645   {
646     \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
647     {
648       \tl_use:N \l__enumext_renew_the_count_vii_tl
649     }
650   }
651   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
652   {
653     \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
654     {
655       \tl_use:N \l__enumext_renew_the_count_viii_tl
656     }
657   }
658 }

```

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

### 11.12.3 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.

```

label
ref
\__enumext_label_v_tl 659 \keys_define:nn { enumext / keyans }
\__enumext_label_vi_tl 660 {
661   label .code:n = {
662     \__enumext_label_style:cnv { \__enumext_label_v_tl }
663     { \__enumext_counter_v_tl } {#1}
664     \dim_set_eq:cN { \__enumext_labelwidth_v_dim }
665     \__enumext_current_widest_dim
666     \__enumext_label_style:cnv { \__enumext_label_vi_tl }
667     { \__enumext_counter_vi_tl } {#1}
668     \dim_set_eq:cN { \__enumext_labelwidth_v_dim }
669     \__enumext_current_widest_dim
670   },
671   label .initial:n = \Alph*,
672   label .value_required:n = true,
673   ref .code:n = \__enumext_keyans_ref:n {#1},
674   ref .value_required:n = true,
675 }

```

(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 676 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
677 {
678   \tl_set:Nn \__enumext_ref_key_arg_tl {#1}
679   \tl_if_empty:NTF \__enumext_ref_key_arg_tl
680   {
681     \msg_error:nnn { enumext } { key-ref-empty } { keyans }
682   }
683   {
684     \tl_set_eq:NN \__enumext_ref_the_count_tl \__enumext_counter_v_tl
685     \__enumext_regex_counter_style:
686     \tl_set_eq:NN \__enumext_ref_the_count_tl \__enumext_the_counter_v_tl
687     \tl_put_right:Ne \__enumext_renew_the_count_v_tl
688     {
689       \exp_not:N \renewcommand { \exp_not:V \__enumext_ref_the_count_tl }
690       { \exp_not:V \__enumext_ref_key_arg_tl }
691     }
692   }
693 }

```

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

```

694 \cs_new_protected:Nn \__enumext_keyans_ref:
695 {
696   \tl_if_empty:NF \__enumext_renew_the_count_v_tl
697   {
698     \tl_use:N \__enumext_renew_the_count_v_tl
699   }
700 }

```

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

### 11.13 Setting start and widest keys

The function `\__enumext_start_from:NNn` used by the **start** key take three arguments:

```

\__enumext_start_from:NNn #1: \__enumext_label_X_tl
\__enumext_start_from:ccn #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.

```

701 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
702 {
703   \__enumext_if_is_int:nTF { #3 }
704   {
705     \int_set:Nn #2 {#3}
706   }

```

```

707     {
708         \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
709         { \int_set:Nn #2 { \int_from_alph:n {#3} } }
710         \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
711         { \int_set:Nn #2 { \int_from_roman:n {#3} } }
712     }
713 }
714 \cs_generate_variant:Nn \__enumext_start_from:NNn { ccn }

```

(End of definition for \\_\_enumext\_start\_from:NNn.)

```

\__enumext_widest_from:nNNn
\__enumext_widest_from:nccn

```

The function \\_\_enumext\_widest\_from:nNNn used by the `widest` key take four arguments:

- #1: The counter associated with the environment level
- #2: \l\_\_enumext\_label\_X\_tl
- #3: \l\_\_enumext\_labelwidth\_X\_dim
- #4: *<integer or string>*

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

```

715 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
716 {
717     \__enumext_if_is_int:nTF {#4}
718     {
719         \setcounter{enumX#1} { #4 }
720     }
721     {
722         \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
723         { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
724         \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
725         { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
726     }
727     \__enumext_label_width_by_box:cv
728     { \l__enumext_labelwidth_#1_dim } { \l__enumext_label_#1_tl }
729 }
730 \cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }

```

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

```

start
widest
\l__enumext_start_X_int

```

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

```

731 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
732 {
733     \keys_define:nn { enumext / #1 }
734     {
735         start .code:n = {
736             \__enumext_start_from:ccn
737             { \l__enumext_label_#2_tl }
738             { \l__enumext_start_#2_int } {##1}
739         },
740         start .initial:n = 1,
741         widest .code:n = {
742             \__enumext_widest_from:nccn {#2}
743             { \l__enumext_label_#2_tl }
744             { \l__enumext_labelwidth_#2_dim } {##1}
745         },
746         widest .value_required:n = true,
747         start .value_required:n = true,
748     }
749 }
750 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

## 11.14 Setting keys for vertical spaces

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

```

topsep
partopsep
parsep
noitemsep
nosep
751 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
752 {
753     \keys_define:nn { enumext / #1 }

```

```

754 {
755   topsep .skip_set:c = { l__enumext_topsep_#2_skip },
756   topsep .initial:n = {#3},
757   topsep .value_required:n = true,
758   partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
759   partopsep .initial:n = {#4},
760   partopsep .value_required:n = true,
761   parsep .skip_set:c = { l__enumext_parsep_#2_skip },
762   parsep .initial:n = {#5},
763   parsep .value_required:n = true,
764   itemsep .skip_set:c = { l__enumext_itemsep_#2_skip },
765   itemsep .initial:n = {#6},
766   itemsep .value_required:n = true,
767   noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
768   noitemsep .value_forbidden:n = true,
769   nosep .meta:n = {
770     itemsep = 0pt, parsep= 0pt,
771     topsep = 0pt, partopsep = 0pt,
772   },
773   nosep .value_forbidden:n = true,
774 }
775 }

```

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

```

776 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
777 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
778 { 4.0pt plus 2.0pt minus 1.0pt }
779 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
780 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
781 { 2.0pt plus 1.0pt minus 1.0pt }
782 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
783 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
784 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
785 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
786 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
787 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
788 { 2.0pt plus 1.0pt minus 1.0pt }
789 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
790 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
791 { 4.0pt plus 2.0pt minus 1.0pt }
792 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
793 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
794 { 2.0pt plus 1.0pt minus 1.0pt }

```

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

### 11.15 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:
795 \cs_set_protected:Npn \__enumext_tmp:n #1
796 {
797   \keys_define:nn { enumext / #1 }
798   {
799     base-fix .bool_set:N = \l__enumext_base_line_fix_bool,
800     base-fix .initial:n = false,
801     base-fix .value_forbidden:n = true,
802   }
803 }
804 \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 (§11.35) and to the function `\__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§11.39)

```

805 \cs_new_protected:Nn \__enumext_nested_base_line_fix:
806 {
807   \bool_lazy_and:nnT

```

```

808 { \bool_if_p:N \__enumext_standar_first_bool }
809 { \bool_if_p:N \__enumext_base_line_fix_bool }
810 {
811   \mode_leave_vertical:
812   \vspace { -\baselineskip }
813   \keys_set:nn { enumext / level-1 }
814   {
815     topsep = 0pt, above = 0pt, above* = 0pt,
816   }
817 }
818 \bool_lazy_and:nnT
819 { \bool_if_p:N \__enumext_starred_first_bool }
820 { \bool_if_p:N \__enumext_base_line_fix_bool }
821 {
822   \mode_leave_vertical:
823   \vspace { -\baselineskip }
824   \keys_set:nn { enumext / enumext* }
825   {
826     topsep = 0pt, above = 0pt, above* = 0pt,
827   }
828 }
829 \bool_set_false:N \__enumext_base_line_fix_bool
830 }

```

🔗 This key is enabled by default in the command `\printkeyans` (§11.42).

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

## 11.16 Setting keys for horizontal spaces

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

```

831 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
832 {
833   \keys_define:nn { enumext / #1 }
834   {
835     itemindent .dim_set:c = { \__enumext_fake_item_indent_#2_dim },
836     itemindent .value_required:n = true,
837     rightmargin .dim_set:c = { \__enumext_rightmargin_#2_dim },
838     rightmargin .value_required:n = true,
839     listparindent .dim_set:c = { \__enumext_listparindent_#2_dim },
840     listparindent .value_required:n = true,
841     list-offset .dim_set:c = { \__enumext_listoffset_#2_dim },
842     list-offset .value_required:n = true,
843     list-indent .code:n =
844       \bool_set_true:c { \__enumext_leftmargin_tmp_#2_bool }
845       \dim_set:cn { \__enumext_leftmargin_tmp_#2_dim } {##1},
846     list-indent .value_required:n = true,
847   }
848 }
849 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__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.

```

850 \cs_set_protected:Npn \__enumext_tmp:n #1
851 {
852   \keys_define:nn { enumext / #1 } { list-indent .initial:n = 0pt, }
853 }
854 \clist_map_inline:nn { enumext*, keyans* } { \__enumext_tmp:n {#1} }

```

### 11.16.1 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 `0pt`. 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.

```

855 \cs_set_protected:Nn \__enumext_fake_item:
856 {
857   \dim_compare:nNnT
858     { \dim_use:c { \__enumext_fake_item_indent_ \__enumext_level: _dim } }
859     >
860     { \c_zero_dim }

```

```

861     {
862         \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
863         {
864             \exp_not:N \mode_leave_vertical:
865             \exp_not:n { \skip_horizontal:n }
866             { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
867             \ignorespaces
868         }
869     }
870 }
871 \cs_set_protected:Nn \__enumext_keyans_fake_item:
872 {
873     \dim_compare:nNnT
874     { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
875     {
876         \tl_set:Ne \l__enumext_fake_item_indent_v_tl
877         {
878             \exp_not:N \mode_leave_vertical:
879             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
880         }
881     }
882 }
883 \cs_set_protected:Nn \__enumext_fake_item_vii:
884 {
885     \dim_compare:nNnT
886     { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
887     {
888         \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
889         {
890             \exp_not:N \mode_leave_vertical:
891             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
892         }
893     }
894 }
895 \cs_set_protected:Nn \__enumext_fake_item_viii:
896 {
897     \dim_compare:nNnT
898     { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
899     {
900         \tl_set:Ne \l__enumext_fake_item_indent_viii_tl
901         {
902             \exp_not:N \mode_leave_vertical:
903             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
904         }
905     }
906 }

```

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

### 11.17 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.

```

907 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
908 {
909     \keys_define:nn { enumext / #1 }
910     {
911         show-length .bool_set:c = { l__enumext_show_length_#2_bool },
912         show-length .initial:n = false,
913     }
914 }
915 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

### 11.18 Setting before, after and first keys

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

```

after
first
916 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2

```



```

917 {
918   \keys_define:nn { enumext / #1 }
919   {
920     before .tl_set:c = { l__enumext_before_no_starred_key_#2_tl },
921     before .value_required:n = true,
922     before* .tl_set:c = { l__enumext_before_starred_key_#2_tl },
923     before* .value_required:n = true,
924     after .tl_set:c = { l__enumext_after_stop_list_#2_tl },
925     after .value_required:n = true,
926     first .tl_set:c = { l__enumext_after_list_args_#2_tl },
927     first .value_required:n = true,
928   }
929 }
930 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

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

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

```

\__enumext_before_args_exec:
\__enumext_before_keys_exec:
\__enumext_after_stop_list:
\__enumext_after_args_exec:
931 \cs_new_protected:Nn \__enumext_before_args_exec:
932 {
933   \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
934 }

```

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

```

935 \cs_new_protected:Nn \__enumext_before_keys_exec:
936 {
937   \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
938 }

```

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

```

939 \cs_new_protected:Nn \__enumext_after_stop_list:
940 {
941   \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
942 }

```

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

```

943 \cs_new_protected:Nn \__enumext_after_args_exec:
944 {
945   \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
946 }

```

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

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

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

```

\__enumext_before_args_exec_v:
\__enumext_before_keys_exec_v:
\__enumext_after_stop_list_v:
\__enumext_after_args_exec_v:
947 \cs_new_protected:Nn \__enumext_before_args_exec_v:
948 {
949   \tl_use:N \l__enumext_before_starred_key_v_tl
950 }

```

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

```

951 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
952 {
953   \tl_use:N \l__enumext_before_no_starred_key_v_tl
954 }

```

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

```

955 \cs_new_protected:Nn \__enumext_after_stop_list_v:
956 {
957   \tl_use:N \l__enumext_after_stop_list_v_tl
958 }

```

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

```

959 \cs_new_protected:Nn \__enumext_after_args_exec_v:
960 {
961   \tl_use:N \l__enumext_after_list_args_v_tl
962 }

```

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

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

```

\__enumext_before_args_exec_vii:
\__enumext_before_keys_exec_vii
\__enumext_after_stop_list_vii:
\__enumext_after_args_exec_vii:

```

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

```

963 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
964 {
965   \tl_use:N \l__enumext_before_starred_key_vii_tl
966 }
967 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
968 {
969   \tl_use:N \l__enumext_before_starred_key_viii_tl
970 }

```

The functions `\__enumext_before_keys_exec_vii:` and `\__enumext_before_keys_exec_viii:` executes the `{⟨code⟩}` set by the `before` key “before” in `enumext*` and `keyans*` environments is started in `{⟨arg two⟩}` of the list. The `{⟨code⟩}` is executed “knowing” all definition and values provides by `⟨keys⟩`.

```

971 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
972 {
973   \tl_use:N \l__enumext_before_no_starred_key_vii_tl
974 }
975 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
976 {
977   \tl_use:N \l__enumext_before_no_starred_key_viii_tl
978 }

```

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

```

979 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
980 {
981   \tl_use:N \l__enumext_after_stop_list_vii_tl
982 }
983 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
984 {
985   \tl_use:N \l__enumext_after_stop_list_viii_tl
986 }

```

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

```

987 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
988 {
989   \tl_use:N \l__enumext_after_list_args_vii_tl
990 }
991 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
992 {
993   \tl_use:N \l__enumext_after_list_args_viii_tl
994 }

```

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

### 11.19 Setting keys for multicols and minipage

mini-env mini-sep columns-sep columns

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.

```

995 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
996 {
997   \keys_define:nn { enumext / #1 }
998   {
999     mini-env .dim_set:c = { \l__enumext_minipage_right_#2_dim },
1000     mini-env .value_required:n = true,
1001     mini-sep .dim_set:c = { \l__enumext_minipage_hsep_#2_dim },
1002     mini-sep .initial:n = 0.3333em,
1003     mini-sep .value_required:n = true,
1004     columns-sep .dim_set:c = { \l__enumext_columns_sep_#2_dim },
1005     columns-sep .value_required:n = true,
1006     columns .int_set:c = { \l__enumext_columns_#2_int },
1007     columns .initial:n = 1,
1008     columns .value_required:n = true,
1009   }
1010 }
1011 \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.

```

1012 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1013 {
1014   \keys_define:nn { enumext / #1 }
1015   {
1016     mini-right .tl_gset:c = { g__enumext_miniright_code_#2_tl },
1017     mini-right .value_required:n = true,
1018     mini-right* .code:n = {
1019       \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
1020       \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1021     },
1022     mini-right* .value_required:n = true,
1023   }
1024 }
1025 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

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

### 11.20 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.

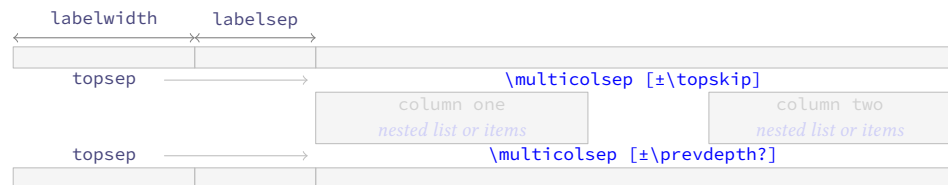


Figure 7: Representation of the vertical space in `multicols` for a nested level.

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

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

#### 11.20.1 Adjustment of vertical spaces for multicols in enumext

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

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

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

```

1026 \cs_new_protected:Nn \__enumext_multi_set_vskip:
1027 {
1028   \skip_set:cn { \l__enumext_multicols_above_ \__enumext_level: _skip }
1029   {
1030     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1031   }
1032   \skip_set:cn { \l__enumext_multicols_below_ \__enumext_level: _skip }
1033   {
1034     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1035   }
1036   \__enumext_add_pre_parsep:
1037 }

```

(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*.

```

1038 \cs_new_protected:Nn \__enumext_add_pre_parsep:
1039 {
1040   \int_case:nn { \l__enumext_level_int }
1041   {
1042     { 2 }{
1043       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1044       {
1045         \skip_add:Nn \l__enumext_multicols_above_ii_skip { \l__enumext_parsep_i_skip }
1046       }
1047     }
1048     { 3 }{
1049       \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1050       {
1051         \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_ii_skip }
1052       }
1053     }
1054     { 4 }{
1055       \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1056       {
1057         \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip }
1058       }
1059     }
1060   }
1061 }

```

(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)*.

```

1062 \cs_new_protected:Nn \__enumext_multi_addvspace:
1063 {
1064   \__enumext_multi_set_vskip:
1065   \mode_if_vertical:T
1066   {
1067     \skip_add:cn { \l__enumext_multicols_above_ \__enumext_level: _skip }
1068     {
1069       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1070     }
1071     \skip_add:cn { \l__enumext_multicols_below_ \__enumext_level: _skip }
1072     {
1073       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1074     }
1075   }
1076   \par\nopagebreak
1077   \addvspace{ \skip_use:c { \l__enumext_multicols_above_ \__enumext_level: _skip } }
1078 }

```

(End of definition for `\__enumext_multi_addvspace:`)

### 11.20.2 Adjustment of vertical spaces for multicol in keyans

`\__enumext_keyans_multi_set_vskip:`  
`\__enumext_keyans_multi_addvspace:`

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

```

1079 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1080 {
1081   \skip_set:Nn \l__enumext_multicol_above_v_skip
1082   {
1083     \l__enumext_topsep_v_skip
1084   }
1085   \skip_set:Nn \l__enumext_multicol_below_v_skip
1086   {
1087     \l__enumext_topsep_v_skip
1088   }
1089 }
1090 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1091 {
1092   \__enumext_keyans_multi_set_vskip:
1093   \mode_if_vertical:T
1094   {
1095     \skip_add:Nn \l__enumext_multicol_above_v_skip
1096     {
1097       \skip_use:N \l__enumext_partopsep_v_skip
1098     }
1099     \skip_add:Nn \l__enumext_multicol_below_v_skip
1100     {
1099       \skip_use:N \l__enumext_partopsep_v_skip
1101     }
1102   }
1103 }
1104 \par\nopagebreak
1105 \addvspace{ \l__enumext_multicol_above_v_skip }
1106 }

```

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

### 11.21 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.

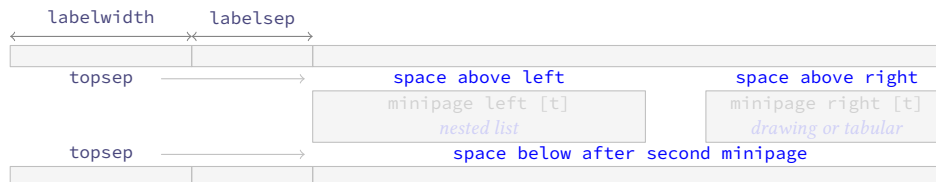


Figure 8: Representation of the `minipage` spacing adjustment for a nested level.

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 `multicol` 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 `multicol` it is nested. The setting of the values was more “trial and error” (aprox to `\strutbox`), using the help of the `lua-visual-debug` package, again my attempts to find the correct values using `\showoutput` and `\showboxdepth` absolutely failed.

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

`\__enumext_mini_set_vskip:`

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

We will set the default values taking into account that `TEX` is in *horizontal mode*, then we will make the settings for the *vertical mode* in which `\partopsep` comes into play.

First determine if the `multicol` environment is active by comparing the value of the `\l__enumext_columns_X_int` variable handled by the `columns` key, according to this comparison we set the adjusted

values for `\l__enumext_minipage_left_skip`, `\l__enumext_minipage_right_skip` and `\l__enumext_minipage_after_skip`.

```

1107 \cs_new_protected:Nn \__enumext_mini_set_vskip:
1108 {
1109   \int_compare:nNnTF
1110   { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1111   {

```

If `multicols` environment is nested in `__enumext_mini_env*` environment, we will apply a correction factor to the *vertical spaces* taking into account the value of `\topsep` of the current level and the value of `\parsep` of the previous level, if these are zero we will use `\strutbox` as the basis for the calculations.

```

1112   \skip_if_eq:nnTF
1113   { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1114   {
1115     \skip_set:Nn \l__enumext_minipage_left_skip
1116     {
1117       -0.150\box_dp:N \strutbox
1118     }
1119     \skip_set:Nn \l__enumext_minipage_right_skip
1120     {
1121       0.695\box_dp:N \strutbox
1122     }
1123     \skip_set:Nn \l__enumext_minipage_after_skip
1124     {
1125       \box_dp:N \strutbox
1126     }
1127     \__enumext_zero_parsep:
1128   }
1129   {
1130     \skip_set:Nn \l__enumext_minipage_left_skip
1131     {
1132       \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1133     }
1134     \skip_set:Nn \l__enumext_minipage_right_skip
1135     {
1136       0.695\box_dp:N \strutbox
1137     }
1138     \skip_set:Nn \l__enumext_minipage_after_skip
1139     {
1140       1.85\box_dp:N \strutbox
1141       + \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1142     }
1143   }
1144 }
1145 {

```

If only `enumext` environment is nested in `__enumext_mini_env*` environment, we will apply a correction factor to the *vertical spaces* taking into account the value of `\topsep`, if this is zero we will use `\strutbox` as the basis for the calculations.

```

1146   \skip_if_eq:nnTF
1147   { \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1148   {
1149     \skip_set:Nn \l__enumext_minipage_left_skip
1150     {
1151       0.5\box_dp:N \strutbox
1152       - \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1153     }
1154     \skip_set:Nn \l__enumext_minipage_right_skip
1155     {
1156       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1157     }
1158     \skip_set:Nn \l__enumext_minipage_after_skip
1159     {
1160       1.6\box_dp:N \strutbox
1161     }
1162   }
1163   {
1164     \skip_set:Nn \l__enumext_minipage_left_skip
1165     {
1166       0.5875\box_dp:N \strutbox
1167       - \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }

```



```

1168         }
1169         \skip_set:Nn \l__enumext_minipage_right_skip
1170         {
1171             + \skip_use:c { \l__enumext_topsep_ \l__enumext_level: _skip }
1172             + \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1173         }
1174         \skip_set:Nn \l__enumext_minipage_after_skip
1175         {
1176             0.325\box_dp:N \strutbox
1177             + \skip_use:c { \l__enumext_topsep_ \l__enumext_level: _skip }
1178         }
1179     }
1180 }
1181 }

```

(End of definition for `\__enumext_mini_set_vskip:`)

`\__enumext_zero_parsep:` The function `\__enumext_zero_parsep:` “adjusted” the value of `\l__enumext_minipage_after_skip` detecting the value of `\parsep` from the previous level. This is necessary since `\parsep` from the previous level affects the *vertical spaces* and this is noticeable when using the `nosep` or `noitemsep` keys.

```

1182 \cs_new_protected:Nn \__enumext_zero_parsep:
1183 {
1184     \int_case:nn { \l__enumext_level_int }
1185     {
1186         { 2 }{
1187             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1188             {
1189                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1190             }
1191         }
1192         { 3 }{
1193             \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1194             {
1195                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1196             }
1197         }
1198         { 4 }{
1199             \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1200             {
1201                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1202             }
1203         }
1204     }
1205 }

```

(End of definition for `\__enumext_zero_parsep:`)

`\__enumext_mini_addvspace:` The function `\__enumext_mini_addvspace:` will apply the spaces set using `\addvspace` “above” the `\__enumext_mini_env*` environment in `enumext`, taking into account whether `TeX` is in *horizontal mode* or *vertical mode*. For the latter we will make some adjustments since the `\partopsep` parameter comes into play and this affects the *vertical spacing*.

```

1206 \cs_new_protected:Nn \__enumext_mini_addvspace:
1207 {
1208     \__enumext_mini_set_vskip:
1209     \mode_if_vertical:T
1210     {
1211         \skip_add:Nn \l__enumext_minipage_left_skip
1212         {
1213             \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1214         }
1215         \skip_add:Nn \l__enumext_minipage_after_skip
1216         {
1217             \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1218         }
1219     }
1220     \par\nopagebreak
1221     \addvspace { \l__enumext_minipage_left_skip }
1222 }

```

(End of definition for `\__enumext_mini_addvspace:`)

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

`\__enumext_keyans_mini_set_vskip:`

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`.

```

1223 \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1224 {
1225   \skip_zero_new:N \l__enumext_minipage_after_skip
1226   \skip_zero_new:N \l__enumext_minipage_left_skip
1227   \skip_zero_new:N \l__enumext_minipage_right_skip
1228   \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1229   {
1230     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1231     {
1232       \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1233       \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1234       \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1235       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1236       {
1237         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1238       }
1239     }
1240     {
1241       \skip_set:Nn \l__enumext_minipage_left_skip
1242       {
1243         \skip_use:N \l__enumext_topsep_v_skip
1244       }
1245       \skip_set:Nn \l__enumext_minipage_right_skip
1246       {
1247         0.705\box_dp:N \strutbox
1248       }
1249       \skip_set:Nn \l__enumext_minipage_after_skip
1250       {
1251         1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1252       }
1253     }
1254   }
1255   {
1256     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1257     {
1258       \skip_set:Nn \l__enumext_minipage_left_skip
1259       {
1260         0.5\box_dp:N \strutbox
1261         + \l__enumext_partopsep_v_skip
1262       }
1263       \skip_set:Nn \l__enumext_minipage_right_skip
1264       {
1265         \l__enumext_partopsep_v_skip
1266       }
1267       \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1268     }
1269     {
1270       \skip_set:Nn \l__enumext_minipage_left_skip
1271       {
1272         0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1273       }
1274       \skip_set:Nn \l__enumext_minipage_right_skip
1275       {
1276         \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1277       }
1278       \skip_set:Nn \l__enumext_minipage_after_skip
1279       {
1280         0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1281       }
1282     }
1283   }
1284 }

```

(End of definition for `\__enumext_keyans_mini_set_vskip:`)

`\__enumext_keyans_mini_addvspace:`

The function `\__enumext_keyans_mini_addvspace:` will apply the spaces set using `\addvspace` “above” the `\__enumext_mini_env*` environment in `keyans`, taking into account whether  $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  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`.

```

1285 \cs_new_protected:Nn \__enumext_keyans_mini_addvspace:
1286 {
1287   \__enumext_keyans_mini_set_vskip:
1288   \mode_if_vertical:T
1289   {
1290     \skip_add:Nn \l__enumext_minipage_left_skip
1291     {
1292       \l__enumext_partopsep_v_skip
1293     }
1294     \skip_add:Nn \l__enumext_minipage_after_skip
1295     {
1296       \l__enumext_partopsep_v_skip
1297     }
1298   }
1299   \par\nopagebreak
1300   \addvspace { \l__enumext_minipage_left_skip }
1301 }

```

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

### 11.21.3 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*`.

```

1302 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1303 {
1304   \skip_zero_new:N \l__enumext_minipage_left_skip
1305   \skip_gzero_new:N \g__enumext_minipage_right_skip
1306   \skip_gzero_new:N \g__enumext_minipage_after_skip
1307   \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1308   {
1309     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1310     \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1311   }
1312   {
1313     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1314     \skip_gset:Nn \g__enumext_minipage_right_skip
1315     {
1316       \l__enumext_topsep_vii_skip
1317     }
1318     \skip_gset:Nn \g__enumext_minipage_after_skip
1319     {
1320       0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1321     }
1322   }
1323 }
1324 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1325 {
1326   \skip_zero_new:N \l__enumext_minipage_after_skip
1327   \skip_zero_new:N \l__enumext_minipage_left_skip
1328   \skip_zero_new:N \l__enumext_minipage_right_skip
1329   \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1330   {
1331     \skip_set:Nn \l__enumext_minipage_left_skip
1332     {
1333       0.5\box_dp:N \strutbox
1334     }
1335     \skip_set:Nn \l__enumext_minipage_right_skip
1336     {
1337       \l__enumext_partopsep_viii_skip
1338     }
1339     \skip_set:Nn \l__enumext_minipage_after_skip
1340     {
1341       1.6\box_dp:N \strutbox
1342     }
1343   }
1344 }

```

```

1345     \skip_set:Nn \l__enumext_minipage_left_skip
1346     {
1347         0.5875\box_dp:N \strutbox
1348     }
1349     \skip_set:Nn \l__enumext_minipage_right_skip
1350     {
1351         \l__enumext_topsep_viii_skip
1352     }
1353     \skip_set:Nn \l__enumext_minipage_after_skip
1354     {
1355         0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1356     }
1357 }
1358 }

```

(End of definition for `\l__enumext_mini_set_vskip_vii:` and `\l__enumext_mini_set_vskip_viii:`)

`\l__enumext_mini_addvspace_vii:`  
`\l__enumext_mini_addvspace_viii:`

The functions `\l__enumext_mini_addvspace_vii:` and `\l__enumext_mini_addvspace_viii:` will apply the vertical space “only above” the `\l__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 *horizontal mode* or *vertical mode*, since `\partopsep` is equal to `0pt` in both environments.

```

1359 \cs_new_protected:Nn \l__enumext_mini_addvspace_vii:
1360 {
1361     \l__enumext_mini_set_vskip_vii:
1362     \par\nopagebreak
1363     \addvspace { \l__enumext_minipage_left_skip }
1364 }
1365 \cs_new_protected:Nn \l__enumext_mini_addvspace_viii:
1366 {
1367     \l__enumext_mini_set_vskip_viii:
1368     \par\nopagebreak
1369     \addvspace { \l__enumext_minipage_left_skip }
1370 }

```

(End of definition for `\l__enumext_mini_addvspace_vii:` and `\l__enumext_mini_addvspace_viii:`)

#### 11.21.4 The command `\miniright`

The command `\miniright` will close the `\l__enumext_mini_env*` environment on the “left side”, open the `\l__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 `\l__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 from being executed inside the `keyanspic` environment, then we call the internal functions for the `enumext` and `keyans` environments.

```

1371 \NewDocumentCommand \miniright { s }
1372 {
1373     \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
1374     {
1375         \msg_error:nnn { enumext } { wrong-miniright-place }
1376     }
1377     \int_compare:nNt { \l__enumext_level_int } = { 0 }
1378     {
1379         \msg_error:nnn { enumext } { wrong-miniright-place }
1380     }
1381     \int_compare:nNtF { \l__enumext_keyans_level_int } = { 1 }
1382     {
1383         \l__enumext_keyans_mini_right_cmd:n {#1}
1384     }
1385     { \l__enumext_mini_right_cmd:n {#1} }
1386 }

```

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

`\l__enumext_mini_right_cmd:n`

The function `\l__enumext_mini_right_cmd:n` takes as argument the *starred* “\*” of the `\miniright` command in the `enumext` environment. We check if the `mini-env` key is active via the variable `\l__enumext_minipage_right_X_dim`, if so we close the `\multicols` environment with the `\l__enumext_mini_env*` environment on the “left side”, then we open the `\l__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.

```

1387 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1388 {
1389   \dim_compare:nNnTF
1390     { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1391     {
1392       \__enumext_multicols_stop:
1393       \end{__enumext_mini_env*}
1394       \hfill
1395       \begin{__enumext_mini_env*}
1396         { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } }
1397         \par\addvspace { \l__enumext_minipage_right_skip }
1398         \bool_if:nF {#1}
1399         {
1400           \centering
1401         }
1402         \int_gzero:N \g__enumext_minipage_stat_int
1403       }
1404       { \msg_error:nnn { enumext } { wrong-miniright-use } }
1405     }

```

(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.

```

1406 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1407 {
1408   \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1409   {
1410     \__enumext_keyans_multicols_stop:
1411     \end{__enumext_mini_env*}
1412     \hfill
1413     \begin{__enumext_mini_env*}{ \l__enumext_minipage_right_v_dim }
1414       \par\addvspace { \l__enumext_minipage_right_skip }
1415       \bool_if:nF {#1}
1416       {
1417         \centering
1418       }
1419       \int_gzero:N \g__enumext_minipage_stat_int
1420     }
1421     { \msg_error:nnn { enumext } { wrong-miniright-use } }
1422   }

```

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

## 11.22 Setting above and below keys

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

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

```

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

```

```

1437     }
1438 }
1439 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

(End of definition for above and others.)

### 11.22.1 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.

```

1440 \cs_new_protected:Nn \__enumext_vspace_above:
1441 {
1442     \skip_if_eq:nnF
1443     { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1444     {
1445         \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
1446         {
1447             \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1448         }
1449         {
1450             \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1451         }
1452     }
1453 }

```

(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.

```

1454 \cs_new_protected:Nn \__enumext_vspace_below:
1455 {
1456     \skip_if_eq:nnF
1457     { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1458     {
1459         \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1460         {
1461             \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1462         }
1463         {
1464             \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1465         }
1466     }
1467 }

```

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

### 11.22.2 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.

```

1468 \cs_new_protected:Nn \__enumext_vspace_above_v:
1469 {
1470     \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1471     {
1472         \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1473         {
1474             \vspace*{ \l__enumext_vspace_above_v_skip }
1475         }
1476         { \vspace { \l__enumext_vspace_above_v_skip } }
1477     }
1478 }

```

(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.

```

1479 \cs_new_protected:Nn \__enumext_vspace_below_v:
1480 {
1481     \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1482     {
1483         \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1484         {
1485             \vspace*{ \l__enumext_vspace_below_v_skip }

```



```

1486     }
1487     { \vspace { \l__enumext_vspace_below_v_skip } }
1488   }
1489 }

```

(End of definition for `\__enumext_vspace_below_v:`)

### 11.22.3 Functions for above and below keys in `enumext*` `keyans*`

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_vii:
\__enumext_vspace_above_viii:
1490 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1491 {
1492   \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1493   {
1494     \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1495     {
1496       \vspace*{ \l__enumext_vspace_above_vii_skip }
1497     }
1498     { \vspace { \l__enumext_vspace_above_vii_skip } }
1499   }
1500 }
1501 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1502 {
1503   \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1504   {
1505     \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1506     {
1507       \vspace*{ \l__enumext_vspace_above_viii_skip }
1508     }
1509     { \vspace { \l__enumext_vspace_above_viii_skip } }
1510   }
1511 }

```

(End of definition for `\__enumext_vspace_above_vii:` and `\__enumext_vspace_above_viii:`)

The functions `\__enumext_vspace_below_vii:` and `\__enumext_vspace_below_viii:` apply the *vertical space below* the `enumext*` and `keyans*` environments set by the `below*` and `below` keys.

```

\__enumext_vspace_below_vii:
\__enumext_vspace_below_viii:
1512 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1513 {
1514   \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1515   {
1516     \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1517     {
1518       \vspace*{ \l__enumext_vspace_below_vii_skip }
1519     }
1520     { \vspace { \l__enumext_vspace_below_vii_skip } }
1521   }
1522 }
1523 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1524 {
1525   \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1526   {
1527     \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1528     {
1529       \vspace*{ \l__enumext_vspace_below_viii_skip }
1530     }
1531     { \vspace { \l__enumext_vspace_below_viii_skip } }
1532   }
1533 }

```

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

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

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

```

series We define the keys series, resume and resume* only for the “first level” of enumext and enumext*.
resume
resume*
1534 \cs_set_protected:Npn \__enumext_tmp:n #1
1535 {
1536   \keys_define:nn { enumext / #1 }
1537   {
1538     series .str_set:N = \__enumext_series_str,
1539     series .value_required:n = true,
1540     resume .code:n = \__enumext_resume_series:n {##1},
1541     resume* .code:n = \__enumext_resume_starred:,
1542     resume* .value_forbidden:n = true,
1543   }
1544 }
1545 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

### 11.23.1 Internal functions for series key

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

```

1546 \cs_new:Npn \__enumext_filter_series:n #1
1547 {
1548   \use:e
1549   {
1550     \keyval_parse:NNn
1551     \__enumext_filter_series_key:n
1552     \__enumext_filter_series_pair:nn {#1}
1553   }
1554 }

```

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

```

1555 \cs_new:Npn \__enumext_filter_series_key:n #1
1556 {
1557   \str_case:nnF {#1}
1558   {
1559     { resume } {}
1560     { resume* } {}
1561     { base-fix } {}
1562   }
1563   { , { \exp_not:n {#1} } }
1564 }

```

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`, `save-ans` and `save-key` keys.

```

1565 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1566 {
1567   \str_case:nnF {#1}
1568   {
1569     { series } {}
1570     { resume } {}
1571     { start } {}
1572     { save-ans } {}
1573     { save-key } {}
1574   }
1575   { , { \exp_not:n {#1} } } = { \exp_not:n {#2} } }
1576 }

```

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

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

- The value of boolean variable `\l__enumext_resume_active_bool` is set to true by the function `\__enumext_resume_counter:n` which is used by the keys `resume` and `resume*`; in this case we must Make sure it is set to false so that it does not overwrite the default filtered *(keys)*. This function is passed to the function `\__enumext_parse_keys:n` in the `enumext` environment definition (§11.35) and to the function `\__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§11.39).

```

1577 \cs_new_protected:Npn \__enumext_parse_series:n #1

```

```

1578 {
1579     \str_if_empty:NTF \l__enumext_series_str
1580     {
1581         \bool_if:NF \l__enumext_resume_active_bool
1582         {
1583             \__enumext_resume_last:n {#1}
1584         }
1585     }
1586     {
1587         \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str_tl }
1588         \tl_gset:ce { g__enumext_series_ \l__enumext_series_str_tl }
1589         { \__enumext_filter_series:n {#1} }
1590         \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str_int }
1591         {
1592             \int_new:c { g__enumext_series_ \l__enumext_series_str_int }
1593         }
1594     }
1595 }

```

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.

```

1596 \cs_new_protected:Npn \__enumext_resume_last:n #1
1597 {
1598     \bool_if:NT \l__enumext_standar_first_bool
1599     {
1600         \tl_gclear:N \g__enumext_standar_series_tl
1601         \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1602     }
1603     \bool_if:NT \l__enumext_starred_first_bool
1604     {
1605         \tl_gclear:N \g__enumext_starred_series_tl
1606         \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1607     }
1608 }

```

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

### 11.23.2 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 (§11.35) and the `enumext*` environment definition (§11.39).

```

1609 \cs_new_protected:Npn \__enumext_resume_save_counter:
1610 {
1611     \bool_if:NT \g__enumext_standar_bool
1612     {
1613         \tl_if_empty:NF \l__enumext_series_str
1614         {
1615             \int_gset_eq:cN
1616             { g__enumext_series_ \l__enumext_series_str_int } \value{enumXi}
1617         }
1618         \tl_if_empty:NTF \l__enumext_resume_name_tl
1619         {
1620             \str_if_empty:NTF \l__enumext_series_str
1621             {
1622                 \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1623             }
1624         }
1625         {
1626             \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl_int }
1627             {

```

```

1628         \int_gset_eq:cN
1629         { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXi}
1630     }
1631 }
1632 \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1633 {
1634     \int_gset_eq:cN
1635     { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1636 }
1637 }
1638 \bool_if:NT \g__enumext_starred_bool
1639 {
1640     \tl_if_empty:NF \l__enumext_series_str
1641     {
1642         \int_gset_eq:cN
1643         { g__enumext_series_ \l__enumext_series_str _int } \value{enumXvii}
1644     }
1645     \tl_if_empty:NTF \l__enumext_resume_name_tl
1646     {
1647         \str_if_empty:NT \l__enumext_series_str
1648         {
1649             \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1650         }
1651     }
1652     {
1653         \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1654         {
1655             \int_gset_eq:cN
1656             { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXvii}
1657         }
1658     }
1659     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1660     {
1661         \int_gset_eq:cN
1662         { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1663     }
1664 }
1665 }

```

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

### 11.23.3 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.

```

1666 \cs_new_protected:Npn \__enumext_resume_series:n #1
1667 {
1668     \tl_if_empty:NTF {#1}
1669     {
1670         \__enumext_resume_counter:n { }
1671     }
1672     {
1673         \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1674         {
1675             \__enumext_resume_counter:n {#1}
1676             \bool_if:NT \g__enumext_standar_bool
1677             {
1678                 \keys_set:nv { enumext / level-1 }
1679                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1680             }
1681             \bool_if:NT \g__enumext_starred_bool
1682             {
1683                 \keys_set:nv { enumext / enumext* }
1684                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1685             }
1686         }
1687     }

```

```

1688         \bool_if:NT \g__enumext_standar_bool
1689         {
1690             \msg_error:nnn { enumext } { unknown-series } {#1}
1691         }
1692         \bool_if:NT \g__enumext_starred_bool
1693         {
1694             \msg_error:nnn { enumext } { unknown-series } {#1}
1695         }
1696     }
1697 }
1698 }

```

(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  $\langle \{series\ name\} \rangle$ . 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=\langle \{series\ name\} \rangle` 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`.

```

1699 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1700 {
1701     \bool_set_true:N \l__enumext_resume_active_bool
1702     \tl_set:Nn \l__enumext_resume_name_tl {#1}
1703     \tl_if_empty:NTF \l__enumext_resume_name_tl
1704     {
1705         \__enumext_resume_counter:
1706     }
1707     {
1708         \__enumext_resume_counter_series:
1709     }
1710     \__enumext_resume_counter_save_ans:
1711 }

```

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.

```

1712 \cs_new_protected:Nn \__enumext_resume_counter:
1713 {
1714     \bool_if:NT \g__enumext_standar_bool
1715     {
1716         \int_gincr:N \g__enumext_resume_int
1717         \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1718     }
1719     \bool_if:NT \g__enumext_starred_bool
1720     {
1721         \int_gincr:N \g__enumext_resume_vii_int
1722         \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1723     }
1724 }

```

The function `\__enumext_resume_counter_series:` will be executed when the `resume=\langle \{series\ name\} \rangle` 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.

```

1725 \cs_new_protected:Nn \__enumext_resume_counter_series:
1726 {
1727     \bool_if:NT \g__enumext_standar_bool
1728     {
1729         \int_set:Nn \l__enumext_start_i_int
1730         {
1731             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1732         }
1733     }
1734     \bool_if:NT \g__enumext_starred_bool
1735     {
1736         \int_set:Nn \l__enumext_start_vii_int
1737         {
1738             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1739         }
1740     }
1741 }

```

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.

```

1742 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1743 {
1744   \bool_lazy_and:nnT
1745     { \bool_if_p:N \__enumext_standar_first_bool }
1746     { \bool_if_p:N \__enumext_store_active_bool }
1747     {
1748       \int_set:Nn \__enumext_start_i_int
1749       {
1750         \int_use:c { g__enumext_resume_ \__enumext_store_name_tl _int } + 1
1751       }
1752     }
1753   \bool_lazy_and:nnT
1754     { \bool_if_p:N \__enumext_starred_first_bool }
1755     { \bool_if_p:N \__enumext_store_active_bool }
1756     {
1757       \int_set:Nn \__enumext_start_vii_int
1758       {
1759         \int_use:c { g__enumext_resume_ \__enumext_store_name_tl _int } + 1
1760       }
1761     }
1762 }

```

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

#### 11.23.4 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.

```

1763 \cs_new_protected:Nn \__enumext_resume_starred:
1764 {
1765   \bool_if:NT \g__enumext_standar_bool
1766   {
1767     \tl_if_empty:NF \g__enumext_standar_series_tl
1768     {
1769       \__enumext_resume_counter:n { }
1770       \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1771     }
1772   }
1773   \bool_if:NT \g__enumext_starred_bool
1774   {
1775     \tl_if_empty:NF \g__enumext_starred_series_tl
1776     {
1777       \__enumext_resume_counter:n { }
1778       \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1779     }
1780   }
1781 }

```

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

#### 11.24 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.

##### 11.24.1 Setting `save-ans` key

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

```

1782 \cs_set_protected:Npn \__enumext_tmp:n #1
1783 {
1784   \keys_define:nn { enumext / #1 }
1785   {
1786     save-ans .code:n = \__enumext_storing_set:n {##1},
1787     save-ans .value_required:n = true,
1788   }
1789 }
1790 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

### 11.24.2 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:`.

```

1791 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
1792 {
1793   \msg_term:nnVV { enumext } { save-ans-log }
1794   \g__enumext_envir_name_tl \l__enumext_store_name_tl
1795 }
1796 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
1797 {
1798   \msg_term:nnVV { enumext } { save-ans-log-hook }
1799   \g__enumext_envir_name_tl \g__enumext_store_name_tl
1800 }

```

(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.

```

1801 \cs_new_protected:Npn \__enumext_storing_set:n #1
1802 {
1803   \tl_set:Nx \l__enumext_store_name_tl {#1}
1804   \tl_if_empty:NTF \l__enumext_store_name_tl
1805   {
1806     \bool_lazy_or:nnT
1807     { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1808     {
1809       \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
1810     }
1811   }
1812   {
1813     \bool_lazy_or:nnT
1814     { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1815     {
1816       \__enumext_start_save_ans_msg:
1817       \__enumext_storing_exec:
1818     }
1819   }
1820 }

```

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*` (§11.28). 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*`.

```

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

```



```

1836     }
1837     \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1838     {
1839         \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
1840         \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1841     }
1842 }

```

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

### 11.24.3 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.

### 11.24.4 Setting check-ans and no-store keys

Now we define the keys `check-ans` and `no-store` for all levels of `enumext` and `enumext*` environments.

```

check-ans
no-store
1843 \cs_set_protected:Npn \__enumext_tmp:n #1
1844 {
1845     \keys_define:nn { enumext / #1 }
1846     {
1847         check-ans .bool_set:N = \l__enumext_check_ans_key_bool,
1848         check-ans .initial:n = false,
1849         check-ans .value_required:n = true,
1850         no-store .code:n = {
1851             \bool_set_false:N \l__enumext_check_answers_bool
1852             \bool_set_false:N \l__enumext_check_ans_key_bool
1853         },
1854         no-store .value_forbidden:n = true,
1855     }
1856 }
1857 \clist_map_inline:nn
1858 {
1859     level-1, level-2, level-3, level-4, enumext*
1860 }
1861 { \__enumext_tmp:n {#1} }

```

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

### 11.24.5 Set-up check answer mechanism

`\__enumext_check_ans_active:` The function `\__enumext_check_ans_active:` will first check the state of the variable `\l__enumext_store_name_tl`, that is, the `save-ans` key is active, if so it will check the state of the variable `\l__enumext_check_answers_bool` handled by the key `no-store` and will execute the function `\__enumext_check_ans_level:` only if “*true*”, i.e. the key `no-store` is not active.

```

1862 \cs_new_protected:Nn \__enumext_check_ans_active:
1863 {
1864   \tl_if_empty:NF \l__enumext_store_name_tl
1865   {
1866     \bool_if:NT \l__enumext_check_answers_bool
1867     {
1868       \__enumext_check_ans_level:
1869     }
1870   }
1871 }

```

The function `\__enumext_check_ans_level:` will decrement by “*one*” the value of the variable `\g__enumext_item_number_int` which keeps track of the executions of `\item` and `\item*` for each level of nesting of the environment `enumext`, taking into account whether it is nested within `enumext*` or the opposite and set `\l__enumext_item_number_bool` to “*false*”.

```

1872 \cs_new_protected:Nn \__enumext_check_ans_level:
1873 {
1874   \int_case:nn { \l__enumext_level_int }
1875   {
1876     { 1 }{
1877       \bool_lazy_all:nT
1878       {
1879         { \bool_if_p:N \g__enumext_starred_bool }
1880         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
1881       }
1882       {
1883         \int_gdecr:N \g__enumext_item_number_int
1884         \bool_set_false:N \l__enumext_item_number_bool
1885       }
1886     }
1887     { 2 }{
1888       \int_gdecr:N \g__enumext_item_number_int
1889       \bool_set_false:N \l__enumext_item_number_bool
1890     }
1891     { 3 }{
1892       \int_gdecr:N \g__enumext_item_number_int
1893       \bool_set_false:N \l__enumext_item_number_bool
1894     }
1895     { 4 }{
1896       \int_gdecr:N \g__enumext_item_number_int
1897       \bool_set_false:N \l__enumext_item_number_bool
1898     }
1899   }

```

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.

```

1900   \int_case:nn { \l__enumext_level_h_int }
1901   {
1902     { 1 }{
1903       \bool_lazy_all:nT
1904       {
1905         { \bool_if_p:N \g__enumext_standar_bool }
1906         { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
1907       }
1908       {
1909         \int_gdecr:N \g__enumext_item_number_int
1910         \bool_set_false:N \l__enumext_item_number_bool
1911       }
1912     }
1913   }
1914 }

```

(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.

```

1915 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
1916 {
1917   \bool_lazy_and:nnT
1918     { \bool_if_p:N \l__enumext_check_ans_key_bool }
1919     { \bool_if_p:N \g__enumext_standar_bool }
1920   {
1921     \bool_gset_true:N \g__enumext_check_ans_key_bool
1922   }
1923   \bool_lazy_and:nnT
1924     { \bool_if_p:N \l__enumext_check_ans_key_bool }
1925     { \bool_if_p:N \g__enumext_starred_bool }
1926   {
1927     \bool_gset_true:N \g__enumext_check_ans_key_bool
1928   }
1929 }

```

(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:.

```

1930 \cs_new_protected:Nn \__enumext_item_answer_diff:
1931 {
1932   \int_gset:Nn \g__enumext_item_answer_diff_int
1933   {
1934     \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
1935   }
1936 }

```

(End of definition for \\_\_enumext\_item\_answer\_diff:.)

\\_\_enumext\_check\_ans\_show:

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:.

```

1937 \cs_new_protected:Nn \__enumext_check_ans_show:
1938 {
1939   \int_case:nn { \g__enumext_item_answer_diff_int }
1940   {
1941     { -1 } { \__enumext_check_ans_msg_less: }
1942     { 0 } { \__enumext_check_ans_msg_same_ok: }
1943     { 1 } { \__enumext_check_ans_msg_greater: }
1944   }
1945 }
1946 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
1947 {
1948   \msg_warning:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
1949   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1950 }
1951 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
1952 {
1953   \msg_term:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
1954   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1955 }
1956 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
1957 {
1958   \msg_warning:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
1959   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1960 }

```

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

\\_\_enumext\_check\_ans\_log:

The function \\_\_enumext\_check\_ans\_log: will be executed within the function \\_\_enumext\_execute\_after\_env: when the key *check-ans* is not active, that is, when \g\_\_enumext\_check\_ans\_key\_bool is “*false*” and write in the log the appropriate message according to the value of \g\_\_enumext\_item\_answer\_diff\_int set by the function \\_\_enumext\_item\_answer\_diff:.

\\_\_enumext\_check\_ans\_log\_msg\_less:

\\_\_enumext\_check\_ans\_log\_msg\_same\_ok:

\\_\_enumext\_check\_ans\_log\_msg\_greater:

```

1961 \cs_new_protected:Nn \__enumext_check_ans_log:
1962 {
1963   \int_case:nn { \g__enumext_item_answer_diff_int }
1964   {
1965     { -1 } { \__enumext_check_ans_log_msg_less: }
1966     { 0 } { \__enumext_check_ans_log_msg_same_ok: }
1967     { 1 } { \__enumext_check_ans_log_msg_greater: }
1968   }
1969 }
1970 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
1971 {
1972   \msg_log:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
1973   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1974 }
1975 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
1976 {
1977   \msg_log:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
1978   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1979 }
1980 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
1981 {
1982   \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
1983   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1984 }

```

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

#### 11.24.6 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.

```

1985 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
1986 {
1987   \int_compare:nNnT
1988   { \g__enumext_check_starred_cmd_int } = { 0 }
1989   {
1990     \msg_warning:nnnV
1991     { enumext } { missing-starred } { #1 } \l__enumext_check_start_line_env_tl
1992   }
1993   \int_compare:nNnT
1994   { \g__enumext_check_starred_cmd_int } > { 1 }
1995   {
1996     \msg_warning:nnnV
1997     { enumext } { many-starred } { #1 } \l__enumext_check_start_line_env_tl
1998   }
1999   \int_gzero:N \g__enumext_check_starred_cmd_int
2000   \tl_clear:N \l__enumext_check_start_line_env_tl
2001 }

```

(End of definition for \\_\_enumext\_check\_starred\_cmd:n.)

#### 11.25 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*` (§11.28) 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.

```

2002 \cs_new_protected:Nn \__enumext_execute_after_env:
2003 {
2004   \int_compare:nNnT { \l__enumext_level_int } = { 0 }
2005   {
2006     \tl_if_empty:NF \g__enumext_store_name_tl
2007     {
2008       \__enumext_stop_save_ans_msg:
2009       \__enumext_item_answer_diff:
2010       \__enumext_log_global_vars:
2011       \__enumext_log_answer_vars:

```

```

2012         \bool_if:NTF \g__enumext_check_ans_key_bool
2013         {
2014             \__enumext_check_ans_show:
2015         }
2016         { \__enumext_check_ans_log: }
2017         \__enumext_undefine_anskey_env:
2018     }
2019     \__enumext_reset_global_vars:
2020 }
2021 }

```

• This function is passed to the function `\__enumext_after_env:nn` for the environments `enumext` (§11.35) and `enumext*` (§11.39) and it is executed only when the environments are not nested or at some level of these..

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

## 11.26 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*`.

```

2022 \cs_set_protected:Npn \__enumext_tmp:n #1
2023 {
2024     \keys_define:nn { enumext / #1 }
2025     {
2026         wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
2027         wrap-ans .initial:n = \fbox{##1},
2028         wrap-ans .value_required:n = true,
2029         wrap-opt .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
2030         wrap-opt .initial:n = [{##1}],
2031         wrap-opt .value_required:n = true,
2032         save-sep .tl_set:N = \l__enumext_store_keyans_item_opt_sep_tl,
2033         save-sep .initial:n = {, ~},
2034         save-sep .value_required:n = true,
2035         mark-ans .tl_set:N = \l__enumext_mark_answer_sym_tl,
2036         mark-ans .initial:n = \textasteriskcentered,
2037         mark-ans .value_required:n = true,
2038         mark-pos .choice:,
2039         mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2040         mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2041         mark-pos / unknown .code:n =
2042             \msg_error:nnee { enumext } { unknown-choice }
2043             { mark-pos } { left, ~ right } { \exp_not:n {##1} },
2044         mark-pos .initial:n = right,
2045         mark-pos .value_required:n = true,
2046         show-ans .bool_set:N = \l__enumext_show_answer_bool,
2047         show-ans .initial:n = false,
2048         show-ans .value_required:n = true,
2049         show-pos .bool_set:N = \l__enumext_show_position_bool,
2050         show-pos .initial:n = false,
2051         show-pos .value_required:n = true,
2052         mark-ref .tl_set:N = \l__enumext_mark_ref_sym_tl,
2053         mark-ref .initial:n = \textasteriskcentered,
2054         mark-ref .value_required:n = true,
2055         save-ref .bool_set:N = \l__enumext_store_ref_key_bool,
2056         save-ref .initial:n = false,
2057         save-ref .value_required:n = true,
2058     }
2059 }
2060 \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`.

```

2061 \cs_set_protected:Npn \__enumext_tmp:n #1
2062 {
2063     \keys_define:nn { enumext / #1 }
2064     {
2065         mark-pos .choice:,
2066         mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2067         mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },

```

```

2068     mark-pos .initial:n = right,
2069     mark-pos .value_required:n = true,
2070     show-ans .bool_set:N = \l__enumext_show_answer_bool,
2071     show-ans .initial:n = false,
2072     show-ans .value_required:n = true,
2073     show-pos .bool_set:N = \l__enumext_show_position_bool,
2074     show-pos .initial:n = false,
2075     show-pos .value_required:n = true,
2076   }
2077 }
2078 \clist_map_inline:nn { keyans, keyans* } { \l__enumext_tmp:n {#1} }

```

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

### 11.26.1 Store optional arguments of the environments

The idea behind “storing” in the *sequence* is to have a copy of the structure of the environment in which the key `save-ans` is being executed so we must capture the optional arguments passed to the levels of the environment in which it is executed and “storing” them.

```

\__enumext_store_active_keys:n
\__enumext_store_active_keys_vii:n

```

The functions `\__enumext_store_active_keys:n` and `\__enumext_store_active_keys_vii:n` will be responsible for “storing” the *keys* filtered from the optional arguments of the environment in which the key `save-ans` is executed and the levels within this for the `enumext` and `enumext*` environments. We will execute this function only if the variable `\l__enumext_store_save_key_X_bool` is false, that is, the key `store-key` is not active, establishing the variable `\l__enumext_store_save_key_X_tl` with the filtered *keys*.

```

2079 \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2080 {
2081   \bool_if:cF { \l__enumext_store_save_key_ \__enumext_level: _bool }
2082   {
2083     \tl_clear:c { \l__enumext_save_key_ \__enumext_level: _tl }
2084     \tl_set:ce
2085       { \l__enumext_store_save_key_ \__enumext_level: _tl }
2086       { \__enumext_filter_save_key:n {#1} }
2087   }
2088 }
2089 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2090 {
2091   \bool_if:NF \l__enumext_store_save_key_vii_bool
2092   {
2093     \tl_clear:N \l__enumext_store_save_key_vii_tl
2094     \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2095   }
2096 }

```

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

### 11.26.2 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 `\l__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.

```

2097 \cs_set_protected:Npn \__enumext_tmp:n #1
2098 {
2099   \keys_define:nn { enumext / enumext* }
2100   {
2101     save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2102     save-key .value_required:n = true,
2103   }
2104   \keys_define:nn { enumext / #1 }
2105   {
2106     save-key .code:n = \__enumext_parse_save_key:n {##1},
2107     save-key .value_required:n = true,
2108   }
2109 }
2110 \clist_map_inline:nn { level-1, level-2, level-3, level-4 } { \l__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 `\l__enumext_store_save_key_X_tl` for `enumext` and `enumext*`.

```

2111 \cs_new_protected:Npn \__enumext_parse_save_key:n #1
2112 {
2113   \bool_set_true:c { \l__enumext_store_save_key_ \__enumext_level: _bool }
2114   \tl_clear:c { \l__enumext_save_key_ \__enumext_level: _tl }
2115   \tl_set:ce
2116     { \l__enumext_store_save_key_ \__enumext_level: _tl }
2117     { \__enumext_filter_save_key:n {#1} }
2118 }
2119 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2120 {
2121   \bool_set_true:N \l__enumext_store_save_key_vii_bool
2122   \tl_clear:N \l__enumext_store_save_key_vii_tl
2123   \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2124 }

```

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

### 11.26.3 Internal functions to store optional arguments

```

\__enumext_filter_save_key:n
  \__enumext_filter_save_key_key:n
  \__enumext_filter_save_key_pair:nn

```

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

```

2125 \cs_new:Npn \__enumext_filter_save_key:n #1
2126 {
2127   \use:e
2128   {
2129     \keyval_parse:NNn
2130       \__enumext_filter_save_key_key:n
2131       \__enumext_filter_save_key_pair:nn {#1}
2132   }
2133 }

```

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.

```

2134 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2135 {
2136   \str_case:nnF {#1}
2137   {
2138     { resume } {} { resume* } {} { no-store } {} { base-fix } {}
2139   }
2140   { , { \exp_not:n {#1} } }
2141 }

```

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.

```

2142 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2143 {
2144   \str_case:nnF {#1}
2145   {
2146     { series } {} { resume } {} { save-ans } {} { save-ref } {}
2147     { save-key } {} { check-ans } {} { show-ans } {} { show-pos } {}
2148     { wrap-ans } {} { mark-ans } {} { wrap-opt } {} { save-sep } {}
2149     { mark-ref } {} { mini-env } {} { mini-sep } {} { mini-right } {}
2150     { mini-right* } {}
2151   }
2152   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2153 }

```

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



#### 11.26.4 Function for storing content in prop list

```
\__enumext_store_addto_prop:n
\__enumext_store_addto_prop:V
```

The function `\__enumext_store_addto_prop:n` stores the content in *⟨prop list⟩* defined by `save-ans` key. The “*stored content*” is retrieved by means of the `\getkeyans` command.

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

```
2154 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2155 {
2156   \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2157   {
2158     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2159   }
2160   { #1 }
2161 }
2162 \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V, e }
```

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

#### 11.26.5 Function for storing content in sequence

```
\__enumext_store_addto_seq:n
\__enumext_store_addto_seq:v
\__enumext_store_addto_seq:V
```

The function `\__enumext_store_addto_seq:n` stores the content in *⟨sequence⟩* defined by `save-ans` key. This function is used by `\anskey` in `enumext`, `\item*` in `keyans` and `\anspic` in `keyanspic`.

The form in which the content is stored in *⟨sequence⟩* is in a internal `enumext` or `enumext*` environments with the *same structure* in which the command was executed.

The “*stored content*” is retrieved by means of the `\printkeyans` command.

```
2163 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2164 {
2165   \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
2166 }
2167 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V, e }
```

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

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

```
\__enumext_store_level_open:
\__enumext_store_level_close:
```

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.

```
2168 \cs_new_protected:Nn \__enumext_store_level_open:
2169 {
2170   \bool_if:NT \l__enumext_check_answers_bool
2171   {
2172     \tl_if_empty:cTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2173     {
2174       \__enumext_store_addto_seq:n
2175       {
2176         \item \begin{enumext}
2177       }
2178     }
2179     {
2180       \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2181       {
2182         \item \begin{enumext} [
2183         }
2184       \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2185       {
2186         ]
2187       }
2188       \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2189     }
2190   }
2191 }
2192 \cs_new_protected:Nn \__enumext_store_level_close:
2193 {
2194   \bool_if:NT \l__enumext_check_answers_bool
2195   {
2196     \__enumext_store_addto_seq:n { \end{enumext} }
2197   }
2198 }
```

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

\\_\_enumext\_store\_level\_open\_vii:  
 \\_\_enumext\_store\_level\_close\_vii:

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.

```

2199 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2200 {
2201   \bool_if:NT \l__enumext_check_answers_bool
2202   {
2203     \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2204     {
2205       \__enumext_store_addto_seq:n
2206       {
2207         \item \begin{enumext*}
2208       }
2209     }
2210     {
2211       \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2212       {
2213         \item \begin{enumext*}[
2214       }
2215       \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2216       {
2217         ]
2218       }
2219       \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2220     }
2221   }
2222 }
2223 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2224 {
2225   \bool_if:NT \l__enumext_check_answers_bool
2226   {
2227     \__enumext_store_addto_seq:n { \end{enumext*} }
2228   }
2229 }

```

(End of definition for \\_\_enumext\_store\_level\_open\_vii: and \\_\_enumext\_store\_level\_close\_vii:.)

### 11.26.7 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

```

2230 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2231 {
2232   \mode_leave_vertical:
2233   \skip_horizontal:n { -\dim_use:N #2 }
2234   \makebox[0pt][ r ]
2235   {
2236     \makebox[ \dim_use:N #1 ] [ \l__enumext_mark_position_str ]
2237     {
2238       \tl_use:N \l__enumext_mark_answer_sym_tl
2239     }
2240   }
2241   \skip_horizontal:n { \dim_use:N #2 }
2242 }
2243 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }

```

(End of definition for \\_\_enumext\_print\_keyans\_box:NN.)

### 11.27 The command \anskey and internal label and ref

Since we will be “*storing content*” in a list environment within *⟨sequences⟩* and can (more or less) manage the options passed to each level, it is necessary that we have a little more control over `\item` when storing.

The `\anskey` command will cover this point and give it similar behaviour to that of `\item` in the `enumext` and `enumext*` environments executed as follows: `\anskey[⟨key = val⟩]{⟨content⟩}` so first we’ll add the keys `break-col`, `item-join`, `item-star`, `item-sym*` and `item-pos*`.

```

2244 \keys_define:nn { enumext / anskey }
2245 {
2246   break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2247   break-col .default:n = true,

```

```

2248     break-col .value_forbidden:n = true,
2249     item-join .int_set:N = \l__enumext_store_item_join_int,
2250     item-join .value_required:n = true,
2251     item-star .bool_set:N = \l__enumext_store_item_star_bool,
2252     item-star .default:n = true,
2253     item-star .value_forbidden:n = true,
2254     item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
2255     item-sym* .value_required:n = true,
2256     item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
2257     item-pos* .value_required:n = true,
2258     unknown .code:n = { \l__enumext_anskey_keys:n {#1} },
2259
2260 }

```

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

```

2261 \cs_new_protected:Npn \__enumext_anskey_keys:n #1
2262 {
2263     \exp_args:NV \__enumext_anskey_keys:nn \l_keys_key_str {#1}
2264 }
2265 \cs_new_protected:Npn \__enumext_anskey_keys:nn #1 #2
2266 {
2267     \tl_if_blank:nTF {#2}
2268     {
2269         \msg_error:nnn { enumext } { anskey-cmd-key-unknown } {#1}
2270     }
2271     {
2272         \msg_error:nnnn { enumext } { anskey-cmd-key-value-unknown } {#1} {#2}
2273     }
2274 }

```

The `\anskey` command will only be present when using the `save-ans` key in `enumext` and `enumext*` environments, otherwise it will return an error.

**\anskey** We will first call the function `\__enumext_anskey_safe_outer:` to be sure where we execute the command, then we will check the state of the variable `\l__enumext_check_answers_bool` set by the key `no-store`, if is true we will increment `\g__enumext_item_anskey_int` for the internal “*check answer*” system and execute the function `\__enumext_anskey_safe_inner:n` to ensure that the command is not nested and that the argument is not empty, finally search the `[<key = val>]` and call the function `\__enumext_store_anskey_code:n`.

```

2275 \NewDocumentCommand \anskey { o +m }
2276 {
2277     \__enumext_anskey_safe_outer:
2278     \group_begin:
2279         \bool_if:NT \l__enumext_check_answers_bool
2280         {
2281             \__enumext_anskey_safe_inner:n {#2}
2282             \tl_if_novalue:nF {#1}
2283             {
2284                 \keys_set:nn { enumext / anskey } {#1}
2285             }
2286             \int_gincr:N \g__enumext_item_anskey_int
2287             \__enumext_store_anskey_code:n {#2}
2288         }
2289     \group_end:
2290 }

```

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

### 11.27.1 Internal functions for the command

**\\_\_enumext\_anskey\_safe\_outer:** The `\__enumext_store_anskey_safe_outer:` function will return the appropriate messages when the command is executed outside the environment in which the `save-ans` key was activated.

**\\_\_enumext\_anskey\_safe\_inner:n**

```

2291 \cs_new_protected:Nn \__enumext_anskey_safe_outer:
2292 {
2293     \bool_if:NF \l__enumext_store_active_bool
2294     {
2295         \msg_error:nnnn { enumext } { anskey-wrong-place } { anskey } { enumext }
2296     }
2297     \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
2298     {
2299         \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans }

```

```

2300     }
2301     \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
2302     {
2303         \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans* }
2304     }
2305     \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
2306     {
2307         \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyanspic }
2308     }
2309 }

```

The `\__enumext_anskey_safe_inner:n` function will first check to see if the passed argument is *empty* and then check to see if the command is nested, if preceded by a not numbered `\item` or if it is in *math mode* returning the appropriate messages.

```

2310 \cs_new_protected:Npn \__enumext_anskey_safe_inner:n #1
2311 {
2312     \tl_if_empty:nT {#1}
2313     {
2314         \msg_error:nn { enumext } { anskey-empty-arg }
2315     }
2316     \int_incr:N \l__enumext_anskey_level_int
2317     \int_compare:nNt { \l__enumext_anskey_level_int } > { 1 }
2318     {
2319         \msg_error:nn { enumext } { anskey-nested }
2320     }
2321     \bool_if:NF \l__enumext_item_number_bool
2322     {
2323         \msg_error:nn { enumext } { anskey-unnumber-item }
2324     }
2325     \mode_if_math:T
2326     {
2327         \msg_error:nne { enumext } { anskey-math-mode } { \c_backslash_str anskey }
2328     }
2329 }

```

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

`\__enumext_store_anskey_code:n`

The internal function `\__enumext_store_anskey_code:n` first we pass the *argument* to the *prop list*, then checks the state of the variable `\l__enumext_store_ref_key_bool` handled by the *save-ref* key and will call the function `\__enumext_store_internal_ref:` for the internal “*label and ref*” system. Followed by this if the *show-ans* or *show-pos* keys are active we will show the “*wrapped*” *argument*.

```

2330 \cs_new_protected:Npn \__enumext_store_anskey_code:n #1
2331 {
2332     \__enumext_store_addto_prop:n {#1}
2333     \bool_if:NT \l__enumext_store_ref_key_bool
2334     {
2335         \__enumext_store_internal_ref:
2336     }
2337     \__enumext_anskey_show_wrap_left:n { #1 }

```

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

```

2338     \tl_clear:N \l__enumext_store_anskey_arg_tl
2339     \bool_lazy_and:nnT
2340     { \bool_if_p:N \l__enumext_store_columns_break_bool }
2341     { \bool_not_p:n { \l__enumext_starred_bool } }
2342     {
2343         \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2344     }
2345     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { \item }

```

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

```

2346     \bool_lazy_and:nnT
2347     { \bool_not_p:n { \l__enumext_starred_bool } }
2348     { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2349     {
2350         \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2351         {
2352             ( \exp_not:V \l__enumext_store_item_join_int )

```

```

2353     }
2354 }

```

And now we will review the keys `item-star`, `item-sym*` and `item-pos*` and pass them to `\l__enumext_store_anskey_arg_tl` along with the `(argument)` for `\anskey` or `(body)` for `anskey*`.

```

2355 \bool_if:NTF \l__enumext_store_item_star_bool
2356 {
2357   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2358   \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2359   {
2360     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2361     {
2362       [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2363     }
2364   }
2365   \dim_compare:nT
2366   {
2367     \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2368   }
2369   {
2370     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2371     {
2372       [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2373     }
2374   }
2375   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2376 }
2377 {
2378   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2379 }

```

Finally we check if the `save-ref` key are active along with the `hyperref` package load, if both conditions are met, it will create the `\hyperlink` with `symbol` set by `mark-ref` key and then store in `(sequence)`.

```

2380 \bool_lazy_and:nnT
2381 { \bool_if_p:N \l__enumext_store_ref_key_bool }
2382 { \bool_if_p:N \l__enumext_hyperref_bool }
2383 {
2384   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2385   {
2386     \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2387     { \exp_not:V \l__enumext_mark_ref_sym_tl }
2388   }
2389 }
2390 \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2391 }

```

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

`\__enumext_store_internal_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. 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`.

```

2392 \cs_new_protected:Nn \__enumext_store_internal_ref:
2393 {
2394   \cs_set_protected:Npn \__enumext_tmp:n ##1
2395   {
2396     \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2397     \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2398     \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2399     \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2400   }
2401   \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2402   \cs_set:Npn \__enumext_tmp:n ##1
2403   { . \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.

```

2404 \bool_lazy_all:nT
2405 {
2406   { \bool_if_p:N \g__enumext_starred_bool }
2407   { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }

```

```

2408     }
2409     {
2410         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2411         { \tl_use:N \l__enumext_label_copy_vii_tl }
2412     }
2413     \bool_lazy_all:nT
2414     {
2415         { \bool_if_p:N \l__enumext_standar_bool }
2416         { \bool_if_p:N \g__enumext_starred_bool }
2417         { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2418     }
2419     {
2420         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2421         {
2422             \tl_use:N \l__enumext_label_copy_vii_tl
2423             \int_step_function:nnN { 1 } { \l__enumext_level_int } \l__enumext_tmp:n
2424         }
2425     }

```

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.

```

2426     \bool_lazy_all:nT
2427     {
2428         { \bool_if_p:N \l__enumext_standar_bool }
2429         { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2430         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2431         { \bool_not_p:n { \l__enumext_starred_bool } }
2432     }
2433     {
2434         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2435         {
2436             \tl_use:N \l__enumext_label_copy_i_tl
2437             \int_step_function:nnN { 2 } { \l__enumext_level_int } \l__enumext_tmp:n
2438         }
2439     }
2440     \cs_set:Npn \l__enumext_tmp:n ##1
2441     { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
2442     \bool_lazy_all:nT
2443     {
2444         { \bool_if_p:N \l__enumext_standar_bool }
2445         { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2446         { \bool_not_p:n { \g__enumext_starred_bool } }
2447         { \int_compare_p:nNn { \l__enumext_level_h_int } > { 0 } }
2448     }
2449     {
2450         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2451         {
2452             \int_step_function:nnN { 1 } { \l__enumext_level_int } \l__enumext_tmp:n
2453             . \tl_use:N \l__enumext_label_copy_vii_tl
2454         }
2455     }

```

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

```

2456     \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2457     {
2458         \l__enumext_store_name_tl \c_colon_str
2459         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2460     }

```

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

```

2461     \tl_put_right:Ne \l__enumext_write_aux_file_tl
2462     {
2463         \l__enumext_newlabel:nn
2464         { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2465         { \l__enumext_newlabel_arg_two_tl }
2466     }
2467     \l__enumext_write_aux_file_tl
2468 }

```

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

`\__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.

```

2469 \cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
2470 {
2471   \par
2472   \bool_if:NT \l__enumext_starred_bool
2473   {
2474     \cs_set:Nn \__enumext_level: { vii }
2475   }
2476   \__enumext_print_keyans_box:cc
2477   { \l__enumext_labelwidth_ \__enumext_level: _dim }
2478   { \l__enumext_labelsep_ \__enumext_level: _dim }
2479   \__enumext_anskey_wrapper:n { #1 }
2480 }

```

(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.

```

2481 \cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
2482 {
2483   \bool_if:NT \l__enumext_show_answer_bool
2484   {
2485     \__enumext_anskey_show_wrap_arg:n { #1 }
2486   }
2487   \bool_if:NT \l__enumext_show_position_bool
2488   {
2489     \tl_set:Nc \l__enumext_mark_answer_sym_tl
2490     {
2491       \group_begin:
2492       \exp_not:N \normalfont
2493       \exp_not:N \footnotesize [ \int_eval:n
2494       {
2495         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2496       }
2497       ]
2498       \group_end:
2499     }
2500     \__enumext_anskey_show_wrap_arg:n { #1 }
2501   }
2502 }

```

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

## 11.28 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  $\langle keys \rangle$  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 an 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:` (§11.25) which is executed after the environment in which the key `save-ans` is active.

```

2503 \cs_new_protected:Nn \__enumext_undefine_anskey_env:
2504 {
2505   \cs_undefine:c { anskey* }
2506   \cs_undefine:c { endanskey* }
2507   \cs_undefine:c { __scontents_anskey*_env_begin: }

```



```

2508 \cs_undefine:c { __scontents_anskey*_env_end: }
2509 }

```

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

```

2510 \__enumext_before_env:nn { enumext }
2511 {
2512   \bool_lazy_and:nnT
2513     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2514     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2515     {
2516       \cs_if_free:cF { __scontents_anskey*_env_begin: }
2517       {
2518         \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2519       }
2520     }
2521 }
2522 \__enumext_before_env:nn { enumext* }
2523 {
2524   \bool_lazy_and:nnT
2525     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2526     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2527     {
2528       \cs_if_free:cF { __scontents_anskey*_env_begin: }
2529       {
2530         \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2531       }
2532     }
2533 }

```

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.

```

2534 \__enumext_before_env:nn { anskey* }
2535 {
2536   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2537   {
2538     \msg_error:nnn { enumext } { anskey-env-wrong } { keyans }
2539   }
2540   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2541   {
2542     \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
2543   }
2544   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2545   {
2546     \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
2547   }
2548   \bool_if:NF \l__enumext_item_number_bool
2549   {
2550     \msg_error:nn { enumext } { anskey-unnumber-item }
2551   }
2552   \mode_if_math:T
2553   {
2554     \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
2555   }
2556 }

```

(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: (§11.24.1)` and we will execute it only if the variable `\l__enumext_anskey_env_bool` is true, with this we prevent it from being executed again when the environment is nested and the key `save-ans` is active, which returns an error for part of the package `scontents`.

```

2557 \cs_new_protected:Npn \__enumext_anskey_env_make:n #1
2558 {
2559   \bool_if:NT \l__enumext_anskey_env_bool
2560   {
2561     \newenvsv{anskey*}[store-env=#1,print-env=false]
2562     \__enumext_anskey_env_exec:
2563   }

```

```

2564 }
2565 \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`.

```

2566 \cs_new_protected:Nn \__enumext_anskey_env_define_keys:
2567 {
2568   \keys_define:nn { scontents / scontents }
2569   {
2570     break-col .bool_gset:N = \g__enumext_store_columns_break_bool,
2571     break-col .default:n   = true,
2572     break-col .value_forbidden:n = true,
2573     item-join .int_gset:N   = \g__enumext_store_item_join_int,
2574     item-join .value_required:n = true,
2575     item-star .bool_gset:N = \g__enumext_store_item_star_bool,
2576     item-star .default:n   = true,
2577     item-star .value_forbidden:n = true,
2578     item-sym* .tl_gset:N   = \g__enumext_store_item_symbol_tl,
2579     item-sym* .value_required:n = true,
2580     item-pos* .dim_gset:N   = \g__enumext_store_item_symbol_sep_dim,
2581     item-pos* .value_required:n = true,
2582     print-env .undefine:,
2583     store-env .undefine:,
2584     write-out .undefine:,
2585     unknown   .code:n      = { \__enumext_anskey_env_keys:n {##1} },
2586   }
2587 }

```

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_keys:n`.

```

2588 \cs_new_protected:Npn \__enumext_anskey_env_keys:n #1
2589 {
2590   \exp_args:NV \__enumext_anskey_env_keys:nn \l_keys_key_str {#1}
2591 }
2592 \cs_new_protected:Npn \__enumext_anskey_env_keys:nn #1#2
2593 {
2594   \tl_if_blank:nTF {#2}
2595   {
2596     \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
2597   }
2598   {
2599     \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
2600   }
2601 }

```

The function `\__enumext_anskey_env_undefine_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`.

```

2602 \cs_new_protected:Nn \__enumext_anskey_env_undefine_keys:
2603 {
2604   \keys_define:nn { scontents / scontents }
2605   {
2606     break-col .undefine:,
2607     item-join .undefine:,
2608     item-star .undefine:,
2609     item-sym* .undefine:,
2610     item-pos* .undefine:,
2611     write-out .code:n = {
2612       \bool_set_false:N \l__scontents_storing_bool
2613       \bool_set_true:N  \l__scontents_writing_bool
2614       \tl_set:Nn \l__scontents_fname_out_tl {##1}
2615     },
2616     write-out .value_required:n = true,
2617     print-env .meta:nn = { scontents } { print-env = ##1 },
2618     print-env .default:n = true,
2619     store-env .meta:nn = { scontents } { store-env = ##1 },
2620     unknown   .code:n = { \__scontents_parse_environment_keys:n {##1} },
2621   }
2622 }

```

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*.

```

2623 \cs_new_protected:Npn \__enumext_rescan_anskey_env:n #1
2624 {
2625   \group_begin:
2626   \int_set:Nn \tex_newlinechar:D { ``^^J }
2627   \__scontents_rescan_tokens:x
2628   {
2629     \endgroup % This assumes \catcode`\=0... Things might go off otherwise.
2630     #1
2631   }
2632 }

```

(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*.

```

2633 \cs_new_protected:Nn \__enumext_anskey_env_exec:
2634 {
2635   \__enumext_before_env:nn { anskey* }
2636   {
2637     \__enumext_anskey_env_define_keys:
2638   }

```

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.

```

2639   \hook_if_empty:nF {env/anskey*/after}
2640   {
2641     \hook_gremove_code:nn {env/anskey*/after} { * }
2642   }
2643   \__enumext_after_env:nn { anskey* }
2644   {
2645     \tl_clear:N \l__enumext_store_anskey_env_tl
2646     \tl_clear:N \l__enumext_store_anskey_opt_tl
2647     \tl_gset:Ne \l__enumext_store_anskey_env_tl
2648     {
2649       \seq_item:ce { g__scontents_name_ \l__enumext_store_name_tl _seq } { -1 }
2650     }
2651     \__enumext_anskey_env_keys:
2652     \__enumext_anskey_env_store:
2653     \__enumext_anskey_env_clean:
2654     \__enumext_anskey_env_undefine_keys:
2655   }
2656 }

```

• 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_undefine_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_keys:` The function `\__enumext_anskey__env_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*.

```

2657 \cs_new_protected:Nn \__enumext_anskey_env_keys:
2658 {
2659   \bool_lazy_and:nnT
2660   { \bool_if_p:N \g__enumext_store_columns_break_bool }
2661   { \bool_not_p:n { \l__enumext_starred_bool } }
2662   {
2663     \tl_put_left:Ne \l__enumext_store_anskey_opt_tl { ,break-col, }
2664   }

```

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

```

2665   \bool_lazy_and:nnT
2666   { \bool_not_p:n { \l__enumext_starred_bool } }
2667   { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }
2668   {
2669     \tl_put_left::Ne \l__enumext_store_anskey_opt_tl

```

```

2670     {
2671     ,item-join = \exp_not:V \g__enumext_store_item_join_int,
2672     }
2673 }

```

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

```

2674 \bool_if:NT \g__enumext_store_item_star_bool
2675 {
2676   \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2677   {
2678     ,item-star,
2679   }
2680   \tl_if_empty:NF \g__enumext_store_item_symbol_tl
2681   {
2682     \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2683     {
2684       ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,
2685     }
2686   }
2687   \dim_compare:nT
2688   {
2689     \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
2690   }
2691   {
2692     \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2693     {
2694       ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
2695     }
2696   }
2697 }
2698 }

```

The function `\__enumext_anskey_env_store:` will be responsible for storing the content of the environment, we will execute and only if the variable `\l__enumext_store_anskey_env_tl` is not empty using the functions `\__enumext_store_anskey_code:n` and `\__enumext_rescan_anskey_env:n`.

```

2699 \cs_new_protected:Nn \__enumext_anskey_env_store:
2700 {
2701   \group_begin:
2702   \tl_if_empty:NF \l__enumext_store_anskey_env_tl
2703   {
2704     \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
2705     {
2706       \exp_args:Ne
2707       \__enumext_store_anskey_code:n
2708       {
2709         \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2710       }
2711     }
2712     {
2713       \keys_set_known:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
2714       \exp_args:Ne
2715       \__enumext_store_anskey_code:n
2716       {
2717         \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2718       }
2719     }
2720   }
2721   \group_end:
2722 }

```

The function `\__enumext_anskey_env_clean:` will return the global variables used by the `⟨keys⟩` to their initial state.

```

2723 \cs_new_protected:Nn \__enumext_anskey_env_clean:
2724 {
2725   \bool_gset_false:N \g__enumext_store_columns_break_bool
2726   \int_gzero:N \g__enumext_store_item_join_int
2727   \bool_gset_false:N \g__enumext_store_item_star_bool
2728   \tl_gclear:N \g__enumext_store_item_symbol_tl
2729   \dim_gzero:N \g__enumext_store_item_symbol_sep_dim
2730 }

```

(End of definition for `\__enumext_anskey_env_keys:`, `\__enumext_anskey_env_store:`, and `\__enumext_anskey_env_clean:`.)

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

### 11.29.1 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`.

```

2731 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2732 {
2733   \tl_clear:N \l__enumext_store_current_label_tl
2734   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2735   {
2736     \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
2737   }
2738   {
2739     \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
2740   }
2741   \tl_if_novalue:nF { #1 }
2742   {
2743     % Set save-sep
2744     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2745     {
2746       \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
2747     }
2748     \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2749   }
2750   \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
2751 }

```

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

### 11.29.2 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.

```

2752 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2753 {
2754   \bool_if:NT \l__enumext_store_ref_key_bool
2755   {
2756     \cs_set_protected:Npn \__enumext_tmp:n ##1
2757     {
2758       \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2759       \tl_reverse:c { l__enumext_label_copy_##1_tl }
2760       \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2761       \tl_reverse:c { l__enumext_label_copy_##1_tl }
2762     }
2763     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2764     \__enumext_keyans_store_ref_aux_i:
2765   }
2766 }

```

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

```

2767 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2768 {
2769   \bool_if:NT \g__enumext_starred_bool
2770   {
2771     \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2772   }

```

```

2773 \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
2774 {
2775   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2776   { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
2777 }
2778 \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
2779 {
2780   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2781   { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2782 }
2783 \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
2784 {
2785   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2786   { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2787 }
2788 \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2789 {
2790   \l__enumext_store_name_tl \c_colon_str
2791   \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2792 }
2793 \__enumext_keyans_store_ref_aux_ii:
2794 }

```

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.

```

2795 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2796 {
2797   \tl_put_right:Ne \l__enumext_write_aux_file_tl
2798   {
2799     \__enumext_newlabel:nn
2800     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2801     { \l__enumext_newlabel_arg_two_tl }
2802   }
2803   \l__enumext_write_aux_file_tl
2804 }

```

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

### 11.29.3 Storing content in sequence

`\__enumext_keyans_addto_seq:n`  
`\__enumext_keyans_addto_seq_link:`

The function `\__enumext_keyans_addto_seq:n` will pass the contents of the current *label* `\l__enumext_label_v_tl` for the *keyans* environment and the `\l__enumext_label_vi_tl` for the *keyanspic* environment when using `\item*` and `\anspic*`, followed by the *contents* of the optional argument of both commands to the `\l__enumext_store_current_label_tl` variable to the sequence defined by the *save-ans* key.

```

2805 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2806 {
2807   \tl_clear:N \l__enumext_store_current_label_tl
2808   \int_compare:nNtF { \l__enumext_keyans_pic_level_int } = { 1 }
2809   {
2810     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
2811   }
2812   {
2813     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
2814   }
2815   \tl_if_novalue:nF { #1 }
2816   {
2817     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2818     {
2819       \tl_put_right:Ne \l__enumext_store_current_label_tl
2820       {
2821         \l__enumext_store_keyans_item_opt_sep_tl
2822       }
2823     }
2824     \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2825   }
2826   \__enumext_keyans_addto_seq_link:
2827 }

```

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.

```

2828 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2829 {
2830   \bool_lazy_and:nnT
2831   { \bool_if_p:N \l__enumext_store_ref_key_bool }
2832   { \bool_if_p:N \l__enumext_hyperref_bool }
2833   {
2834     \tl_put_right:Ne \l__enumext_store_current_label_tl
2835     {
2836       \hfill \exp_not:N \hyperlink
2837       {
2838         \exp_not:V \l__enumext_newlabel_arg_one_tl
2839       }
2840       { \exp_not:V \l__enumext_mark_ref_sym_tl }
2841     }
2842   }
2843   \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
2844   \bool_if:NT \l__enumext_check_answers_bool
2845   {
2846     \int_gincr:N \g__enumext_item_anskey_int
2847   }
2848 }

```

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

#### 11.29.4 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.

```

2849 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
2850 {
2851   \tl_if_novalue:nF { #1 }
2852   {
2853     \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
2854   }
2855   \bool_if:NT \l__enumext_show_answer_bool
2856   {
2857     \__enumext_keyans_show_ans:
2858   }
2859   \bool_if:NT \l__enumext_show_position_bool
2860   {
2861     \__enumext_keyans_show_pos:
2862   }
2863 }
2864 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
2865 {
2866   \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
2867   {
2868     \bool_lazy_or:nnT
2869     { \bool_if_p:N \l__enumext_show_answer_bool }
2870     { \bool_if_p:N \l__enumext_show_position_bool }
2871     {
2872       \__enumext_keyans_wrapper_opt:n { \l__enumext_store_current_opt_arg_tl } \c_space_tl
2873     }
2874   }
2875 }
2876 \cs_new_protected:Nn \__enumext_keyans_show_ans:
2877 {
2878   \tl_put_left:Nn \l__enumext_label_v_tl
2879   {
2880     \__enumext_print_keyans_box:NN
2881     \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2882   }
2883 }
2884 \cs_new_protected:Nn \__enumext_keyans_show_pos:
2885 {
2886   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }

```



```

2887     {
2888         \tl_set:Nc \l__enumext_mark_answer_sym_tl
2889         {
2890             \group_begin:
2891             \exp_not:N \normalfont
2892             \exp_not:N \footnotesize [ \int_eval:n
2893             {
2894                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2895             }
2896             ]
2897             \group_end:
2898         }
2899     }
2900     {
2901         \tl_set:Nc \l__enumext_mark_answer_sym_tl
2902         {
2903             \group_begin:
2904             \exp_not:N \normalfont
2905             \exp_not:N \footnotesize [ \int_eval:n
2906             {
2907                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
2908             }
2909             ]
2910             \group_end:
2911         }
2912     }
2913     \tl_put_left:Nn \l__enumext_label_v_tl
2914     {
2915         \__enumext_print_keyans_box:NN
2916         \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2917     }
2918 }

```

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

### 11.30 Setting `item-sym*` and `item-pos*` keys

In order to have a cleaner implementation of `\item*` 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*
2919 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2920 {
2921     \keys_define:nn { enumext / #1 }
2922     {
2923         item-sym* .tl_set:c = { l__enumext_item_symbol_#2_tl },
2924         item-sym* .value_required:n = true,
2925         item-sym* .initial:n = { $\star$ },
2926         item-pos* .dim_set:c = { l__enumext_item_symbol_sep_#2_dim },
2927         item-pos* .value_required:n = true,
2928     }
2929 }
2930 \clist_map_inline:nn
2931 {
2932     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
2933 }
2934 { \__enumext_tmp:nn #1 }

```

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

### 11.31 Redefining `\footnote` command

`\__enumext_footnotetext:nn` To keep the correct numbering of `\footnote` and to make it work correctly with the `mini-env` key and 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](#).

```

2935 \cs_new_protected:Nn \__enumext_footnotetext:nn
2936 {
2937     \footnotetext[#1]{#2}
2938 }
2939 \cs_new_protected:Nn \__enumext_renew_footnote:
2940 {
2941     \seq_gclear:N \g__enumext_footnote_arg_seq

```

```

2942 \seq_gclear:N \g__enumext_footnote_int_seq
2943 \RenewDocumentCommand \footnote { o +m }
2944 {
2945   \tl_if_novalue:nTF {##1}
2946   {
2947     \stepcounter{footnote}
2948     \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
2949   }
2950   {
2951     \int_gset:Nn \g__enumext_footnote_int { ##1 }
2952   }
2953   \footnotemark [ \g__enumext_footnote_int ]
2954   \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
2955   \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
2956 }
2957 }
2958 \cs_new_protected:Nn \__enumext_print_footnote:
2959 {
2960   \seq_if_empty:NF \g__enumext_footnote_int_seq
2961   {
2962     \seq_map_pairwise_function:NNN
2963     \g__enumext_footnote_int_seq
2964     \g__enumext_footnote_arg_seq
2965     \__enumext_footnotetext:nn
2966   }
2967 }

```

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

## 11.32 Redefining `\item` command

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.

### 11.32.1 The `\item` command in enumext

`\__enumext_default_item:n`

The `\item` and `\item[custom]` commands work in the usual way on `enumext`.

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_ans_key_bool` set by the key `check-ans`, set the boolean variable `\l__enumext_wrap_label_X_bool` to “true” and execute `\__enumext_item_std:w`.

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.

The boolean variable `\l__enumext_wrap_label_X_bool` is used by the function `\__enumext_make_label:` (§11.33).

```

2968 \cs_new_protected:Npn \__enumext_default_item:n #1
2969 {
2970   \tl_if_novalue:nTF {#1}
2971   {
2972     \bool_if:NT \l__enumext_check_answers_bool
2973     {
2974       \int_gincr:N \g__enumext_item_number_int
2975       \bool_set_true:N \l__enumext_item_number_bool
2976     }
2977     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2978     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2979   }
2980   {
2981     \bool_set_eq:cc
2982     { l__enumext_wrap_label_ \__enumext_level: _bool }
2983     { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2984     \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2985   }
2986 }

```

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

`\__enumext_starred_item:nn`

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 set by the `labelsep` key and can be *offset* using 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 optional argument in the global variable `\g__enumext_item_symbol_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_ans_key_bool` set by the key `check-ans`, set the boolean variable `\l__enumext_wrap_label_X_bool` to “true” and execute `\__enumext_item_std:w`.

In this function the optional argument of `\__enumext_item_std:w` is omitted, we only want it to be numbered.

The boolean variable `\l__enumext_wrap_label_X_bool` and the vars `\l__enumext_item_symbol_sep_X_dim`, `\g__enumext_item_symbol_tl` are used by the function `\__enumext_make_label:` (§11.33).

```

2987 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2988 {
2989   \tl_if_novalue:nF {#1}
2990   {
2991     \tl_set:cn { \__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2992   }
2993   \tl_gset_eq:Nc \g__enumext_item_symbol_tl { \__enumext_item_symbol_ \__enumext_level: _tl }
2994   \tl_if_novalue:nTF {#2}
2995   {
2996     \dim_set_eq:cc
2997     { \__enumext_item_symbol_sep_ \__enumext_level: _dim }
2998     { \__enumext_labelsep_ \__enumext_level: _dim }
2999   }
3000   {
3001     \dim_set:cn { \__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
3002   }
3003   \bool_if:NT \l__enumext_check_answers_bool
3004   {
3005     \int_gincr:N \g__enumext_item_number_int
3006     \bool_set_true:N \l__enumext_item_number_bool
3007   }
3008   \bool_set_true:c { \__enumext_wrap_label_ \__enumext_level: _bool }
3009   \__enumext_item_std:w \tl_use:c { \__enumext_fake_item_indent_ \__enumext_level: _tl }
3010 }

```

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

`\__enumext_redefine_item:`

The function `\__enumext_redefine_item:` will redefine the `\item` command in the `enumext` environment for the internal mechanism of check-answers for `check-ans` key and adding the starred `\item*` version.

This function is passed to `\__enumext_list_arg_two_X:` which is used in the definition of the `enumext` environment (§11.34.2).

```

3011 \cs_new_protected:Nn \__enumext_redefine_item:
3012 {
3013   \RenewDocumentCommand \item { s o o }
3014   {
3015     \bool_if:nTF {##1}
3016     {
3017       \__enumext_starred_item:nn {##2} {##3}
3018     }
3019     { \__enumext_default_item:n {##2} }
3020   }
3021 }

```

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

### 11.32.2 The `\item` command in keyans

The `\item*` and `\item*[\langle content \rangle]` commands *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.

`\__enumext_keyans_default_item:n`

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

```

3022 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
3023 {
3024   \tl_if_novalue:nTF { #1 }
3025   {

```

```

3026         \bool_set_true:N \l__enumext_wrap_label_v_bool
3027         \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
3028     }
3029     {
3030         \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
3031         \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
3032     }
3033 }

```

(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>`.

```

3034 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
3035 {
3036     \tl_set_eq:NN \l__enumext_store_current_label_tmp_tl \l__enumext_label_v_tl
3037     \__enumext_keyans_show_left:n { #1 }
3038     \bool_set_true:N \l__enumext_wrap_label_v_bool
3039     \__enumext_item_std:w \tl_use:N \l__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>`.

```

3040     \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_store_current_label_tmp_tl
3041     \__enumext_keyans_addto_prop:n { #1 }
3042     \__enumext_keyans_store_ref:
3043     \__enumext_keyans_addto_seq:n { #1 }
3044     \int_gincr:N \g__enumext_check_starred_cmd_int
3045 }

```

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

`\item*`  
`\__enumext_keyans_redefine_item:`

The function `\__enumext_keyans_redefine_item:` is responsible for adding the *starred* and *optional* argument by the `\__enumext_list_arg_two_v:` function in the definition of the `keyans` environment. Here we need to use `\peek_remove_spaces:n` to prevent an unwanted space when using `\item*` in conjunction with the `itemindent` key.

This function is passed to `\__enumext_list_arg_two_v:` which is used in the definition of the `keyans` environment (§11.34.2).

```

3046 \cs_new_protected:Npn \__enumext_keyans_redefine_item:
3047 {
3048     \RenewDocumentCommand \item { s o }
3049     {
3050         \bool_if:nTF {##1}
3051         {
3052             \peek_remove_spaces:n
3053             {
3054                 \__enumext_keyans_starred_item:n {##2}
3055             }
3056         }
3057         {
3058             \__enumext_keyans_default_item:n {##2}
3059         }
3060     }
3061 }

```

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

### 11.33 Redefining `\makelabel` command

Redefine `\makelabel` for the keys `align`, `font`, `wrap-label`, `wrap-label*` and `\item*` for `enumext` and `keyans` environments.

### 11.33.1 Redefining \makeLabel for enumext

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

```

3062 \cs_new_protected:Nn \__enumext_item_starred:
3063 {
3064   \tl_if_empty:cF { \__enumext_item_symbol_ \__enumext_level: _tl }
3065   {
3066     \mode_leave_vertical:
3067     \skip_horizontal:n { -\dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3068     \makebox[0pt][r]{ \g__enumext_item_symbol_tl }
3069     \skip_horizontal:n { \dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3070   }
3071 }

```

(End of definition for `\__enumext_item_starred:`)

`\__enumext_make_label:` The function `\__enumext_make_label:` redefine `\makeLabel` for the `enumext` environment. This function is passed to `\__enumext_list_arg_two_X:` which is used in the definition of the `enumext` environment (§11.34.2).

```

3072 \cs_new_protected:Nn \__enumext_make_label:
3073 {
3074   \RenewDocumentCommand \makeLabel { m }
3075   {
3076     \tl_use:c { \__enumext_label_fill_left_ \__enumext_level: _tl }
3077     \tl_use:c { \__enumext_label_font_style_ \__enumext_level: _tl }
3078     \bool_if:cTF { \__enumext_wrap_label_ \__enumext_level: _bool }
3079     {
3080       \__enumext_item_starred:
3081       \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3082     }
3083     { ##1 }
3084     \tl_use:c { \__enumext_label_fill_right_ \__enumext_level: _tl }
3085     \tl_gclear:N \g__enumext_item_symbol_tl
3086   }
3087 }

```

(End of definition for `\__enumext_make_label:`)

### 11.33.2 Redefining \makeLabel for keyans

`\__enumext_keyans_make_label:` The function `\__enumext_keyans_make_label:` redefine `\makeLabel` for `keyans` environment. This function is passed to `\__enumext_list_arg_two_v:` which is used in the definition of the `keyans` environment (§11.34.2).

```

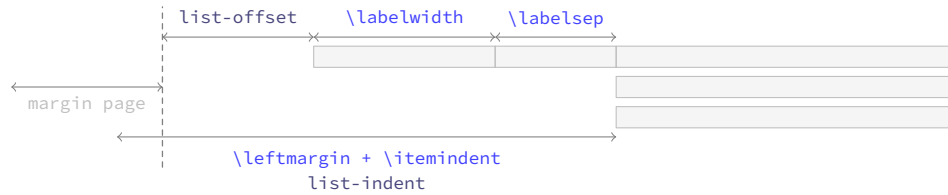
3088 \cs_new_protected:Nn \__enumext_keyans_make_label:
3089 {
3090   \RenewDocumentCommand \makeLabel { m }
3091   {
3092     \tl_use:N \l__enumext_label_fill_left_v_tl
3093     \tl_use:N \l__enumext_label_font_style_v_tl
3094     \bool_if:NTF \l__enumext_wrap_label_v_bool
3095     {
3096       \__enumext_wrapper_label_v:n { ##1 }
3097     }
3098     { ##1 }
3099     \tl_use:N \l__enumext_label_fill_right_v_tl
3100   }
3101 }

```

(End of definition for `\__enumext_keyans_make_label:`)

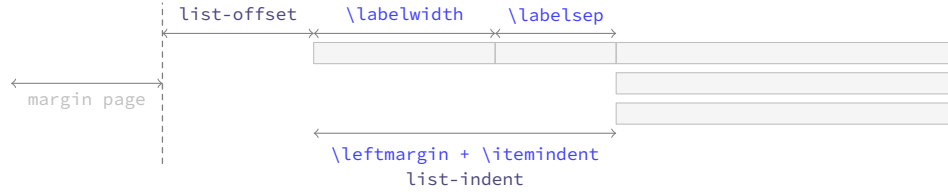
## 11.34 Second argument of the lists

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`.

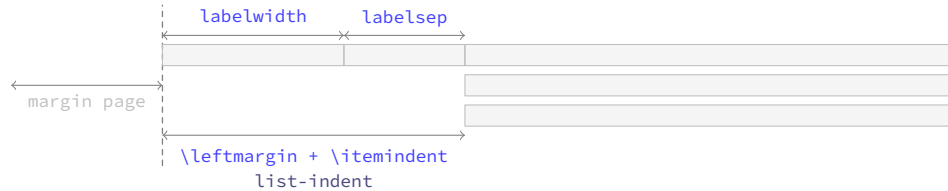
Figure 9: Representation of standard horizontal lengths in `list` environment.

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

Consider the figure 9 where the default margins (on the left) of a list are represented. The idea is to have control over these margins so that our list does not overlap the left margin of the page. The key relationship is that the right edge of the `\labelsep` equals the right edge of the `\itemindent`, so that the left edge of the *label box* is at `\leftmargin + \itemindent` minus `\labelwidth + \labelsep`. Thus, the handling of the margins by the package will be as shown in the figure 10.

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

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

Figure 11: Default horizontal lengths in `enumext`.

```
\__enumext_calc_hspace:NNNNNNN
\__enumext_calc_hspace:ccccccc
```

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

```
#1: \__enumext_labelwidth_X_dim      #2: \__enumext_labelsep_X_dim
#3: \__enumext_listoffset_X_dim      #4: \__enumext_leftmargin_tmp_X_dim
#5: \__enumext_leftmargin_X_dim      #6: \__enumext_itemindent_X_dim
#7: \__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 (§11.34.2).

```
3102 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
3103 {
3104   \dim_compare:nNt { #1 } < { \c_zero_dim }
3105   {
3106     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
3107     \dim_set:Nn #1 { \dim_abs:n { #1 } }
3108   }
3109   \dim_compare:nNt { #2 } < { \c_zero_dim }
3110   {
3111     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
3112     \dim_set:Nn #2 { \dim_abs:n { #2 } }
3113   }
3114 }
```

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

```
3114 \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }
```

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

```
3115 \dim_compare:nNtF { #4 } < { \c_zero_dim }
3116 {
3117   \dim_set:Nn #6 { #1 + #2 - #4 }
3118   \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3119 }
```

```

3120 {
3121   \dim_compare:nNnT { #4 } = { #1 + #2 }
3122   { \dim_set:Nn #6 { \c_zero_dim } }
3123   \dim_compare:nNnT { #4 } < { #1 + #2 }
3124   { \dim_set:Nn #6 { #1 + #2 - #4 } }
3125   \dim_compare:nNnT { #4 } > { #1 + #2 }
3126   {
3127     \dim_set:Nn #6 { -#1 - #2 + #4 }
3128     \dim_set:Nn #6 { #6*-1 }
3129   }
3130   \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3131 }
3132 }
3133 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { cccccc }

```

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

### 11.34.2 Setting second argument of the lists

We will “not set” `\leftmargini`, `\leftmarginii`, `\leftmarginiii` or `\leftmarginiv`, in this case, we will directly set the parameters for vertical and horizontal list spacing per level.

```

\__enumext_list_arg_two_i:
\__enumext_list_arg_two_ii:
\__enumext_list_arg_two_iii:
\__enumext_list_arg_two_iv:
\__enumext_list_arg_two_v:
3134 \cs_set_protected:Npn \__enumext_tmp:n #1
3135 {
3136   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3137   {
3138     \__enumext_calc_hspace:ccccc
3139     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3140     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3141     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3142     { \__enumext_leftmargin_tmp_#1_bool }
3143     \clist_map_inline:nn
3144     { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3145     { \dim_set_eq:cc {###1} { \__enumext_###1_#1_dim } }
3146     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3147     { \skip_set_eq:cc {###1} { \__enumext_###1_#1_skip } }
3148     \usecounter { enumX#1 }
3149     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
3150     \str_if_eq:nnTF {#1} { v }
3151     {
3152       \__enumext_keyans_redefine_item:
3153       \__enumext_keyans_make_label:
3154       \__enumext_keyans_ref:
3155       \__enumext_keyans_fake_item:
3156       \bool_if:cT { \__enumext_show_length_#1_bool }
3157       {
3158         \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3159       }
3160     }
3161     {
3162       \__enumext_redefine_item:
3163       \__enumext_make_label:
3164       \__enumext_standar_ref:
3165       \__enumext_fake_item:
3166       \bool_if:cT { \__enumext_show_length_#1_bool }
3167       {
3168         \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \__enumext_level_#1_int }
3169       }
3170     }
3171   }
3172 }
3173 \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 `lisparindent` and `parsep` to set the value of `\parskip` locally.

```

3174 \cs_set_protected:Npn \__enumext_tmp:n #1
3175 {
3176   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3177   {

```



```

3178 \__enumext_calc_hspace:ccccc
3179 { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3180 { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3181 { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3182 { \__enumext_leftmargin_tmp_#1_bool }
3183 \clist_map_inline:nn
3184 { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3185 { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
3186 \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3187 { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
3188 \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
3189 \skip_zero:N \partopsep
3190 \usecounter { enumX#1 }
3191 \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
3192 \__enumext_starred_ref:
3193 \str_if_eq:nnTF {#1} { vii }
3194 {
3195     \__enumext_fake_item_vii:
3196     \bool_if:cT { \__enumext_show_length_vii_bool }
3197     { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3198 }
3199 {
3200     \__enumext_fake_item_viii:
3201     \bool_if:cT { \__enumext_show_length_#1_bool }
3202     { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3203 }
3204 }
3205 }
3206 \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:.)

## 11.35 The environment enumext

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

```

3207 \NewDocumentEnvironment{enumext}{0}{ }
3208 {
3209     \__enumext_safe_exec:
3210     \__enumext_parse_keys:n {#1}
3211     \__enumext_before_list:
3212     \__enumext_start_store_level:
3213     \__enumext_start_list:nn
3214     { \tl_use:c { \__enumext_label_ \__enumext_level: _tl } }
3215     {
3216         \use:c { __enumext_list_arg_two_ \__enumext_level: : }
3217         \__enumext_before_keys_exec:
3218     }
3219     \__enumext_after_args_exec:
3220 }
3221 {
3222     \__enumext_stop_list:
3223     \__enumext_stop_store_level:
3224     \__enumext_after_list:
3225 }

```

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

`\__enumext_safe_exec:` The `\__enumext_safe_exec:` function first execute the function `\__enumext_is_not_nested:` which will set the variable `\__enumext_standar_bool` to “true” if the environment is not nested in `enumext*`, we increment the variable `\__enumext_level_int` for the nesting levels and set the `\__enumext_standar_bool` variable to “true”. Finally we set the variable `\__enumext_standar_first_bool` to “true” only if the environment is not nested and we are at the “first level” of it using the function `\__enumext_is_on_first_level:`.

```

3226 \cs_new_protected:Nn \__enumext_safe_exec:
3227 {
3228     \__enumext_internal_mini_page:
3229     \__enumext_is_not_nested:
3230     \int_incr:N \__enumext_level_int
3231     \int_compare:nNt { \__enumext_level_int } > { 4 }
3232     { \msg_fatal:nn { enumext } { list-too-deep } }
3233     \bool_set_true:N \__enumext_standar_bool

```

```

3234     \__enumext_is_on_first_level:
3235 }

```

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

\\_\_enumext\_parse\_keys:n

The \\_\_enumext\_parse\_store\_keys:n function will parse the *⟨keys⟩* passed to the optional environment argument *enumext* by levels only if present. First we clear the variable \l\_\_enumext\_series\_str 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*, otherwise we will pass the *⟨keys⟩* to the inner levels of the environment and finally if the variable \l\_\_enumext\_store\_active\_bool established by the key *save-ans* is true we execute \\_\_enumext\_parse\_store\_keys:n used by the key *save-key*.

```

3236 \cs_new_protected:Npn \__enumext_parse_keys:n #1
3237 {
3238     \tl_if_no_value:nF {#1}
3239     {
3240         \str_clear:N \l__enumext_series_str
3241         \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
3242         {
3243             \keys_set:nn { enumext / level-1 } {#1}
3244             \__enumext_parse_series:n {#1}
3245             \__enumext_nested_base_line_fix:
3246         }
3247         {
3248             \exp_args:Ne \keys_set:nn
3249             { enumext / level-\int_use:N \l__enumext_level_int } {#1}
3250         }
3251         \__enumext_store_active_keys:n {#1}
3252     }
3253 }

```

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

\\_\_enumext\_start\_store\_level:

\\_\_enumext\_stop\_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\*.

```

3254 \cs_new_protected:Nn \__enumext_start_store_level:
3255 {
3256     \bool_lazy_all:nT
3257     {
3258         { \bool_if_p:N \l__enumext_store_active_bool }
3259         { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3260         { \bool_if_p:N \g__enumext_standar_bool }
3261     }
3262     {
3263         \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3264         {
3265             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3266             \__enumext_store_level_open:
3267         }
3268     }

```

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

```

3269     \bool_lazy_all:nT
3270     {
3271         { \bool_if_p:N \l__enumext_store_active_bool }
3272         { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3273         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
3274     }
3275     {
3276         \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3277         {
3278             \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3279             \__enumext_store_level_open:
3280         }
3281     }
3282 }
3283 \cs_new_protected:Nn \__enumext_stop_store_level:
3284 {
3285     \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
3286     {
3287         \__enumext_store_level_close:

```

```

3288     }
3289 }

```

(End of definition for `\__enumext_start_store_level:` and `\__enumext_stop_store_level:`.)

`\__enumext_before_list:` The function `\__enumext_before_list:` 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.

```

3290 \cs_new_protected:Nn \__enumext_before_list:
3291 {
3292     \__enumext_vspace_above:
3293     \__enumext_before_args_exec:

```

The function `\__enumext_check_ans_active:` will handle the check answer mechanism, which will be activated with the `check-ans` key.

```

3294     \__enumext_check_ans_active:

```

When the `mini-env` key is active it will set the value of the `\__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 `\__enumext_minipage_hsep_X_dim` set by the `mini-sep` key, the value of `\__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.

```

3295     \dim_compare:nNt
3296     { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3297     {
3298         \dim_set:cn { \__enumext_minipage_left_ \__enumext_level: _dim }
3299         {
3300             \linewidth
3301             - \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim }
3302             - \dim_use:c { \__enumext_minipage_hsep_ \__enumext_level: _dim }
3303         }

```

The boolean variable `\__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_mini_addvspace:` 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.

Here we use the plain TeX macro `\nointerlineskip` to prevent baseline “glue” being added between the next pair of boxes in a *vertical list*.

```

3304         \bool_set_true:c { \__enumext_minipage_active_ \__enumext_level: _bool }
3305         \int_gincr:N \g__enumext_minipage_stat_int
3306         \__enumext_mini_addvspace:
3307         \nointerlineskip\noindent
3308         \begin{\__enumext_mini_env*}
3309         { \dim_use:c { \__enumext_minipage_left_ \__enumext_level: _dim } }
3310     }
3311     \__enumext_multicols_start:
3312 }

```

(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=0pt` and set the value of `\multicolsep` equal to zero and leave `\columnseprule` equal to zero for inner levels.

```

3313 \cs_new_protected:Nn \__enumext_multicols_start:
3314 {
3315     \int_compare:nNt
3316     { \int_use:c { \__enumext_columns_ \__enumext_level: _int } } > { 1 }
3317     {
3318         \dim_compare:nNt
3319         { \dim_use:c { \__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3320         {
3321             \dim_set:cn { \__enumext_columns_sep_ \__enumext_level: _dim }
3322             {
3323                 ( \dim_use:c { \__enumext_labelwidth_ \__enumext_level: _dim }
3324                   + \dim_use:c { \__enumext_labelsep_ \__enumext_level: _dim }
3325                 ) / \int_use:c { \__enumext_columns_ \__enumext_level: _int }
3326                 - \dim_use:c { \__enumext_listoffset_ \__enumext_level: _dim }
3327             }
3328         }

```

```

3329     \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
3330     \skip_zero:N \multicolsep
3331     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3332     {
3333         \dim_zero:N \columnseprule
3334     }

```

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.

```

3335     \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3336     {
3337         \__enumext_multi_addvspace:
3338     }
3339     \raggedcolumns
3340     \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
3341 }
3342 }

```

(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.

```

3343 \cs_new_protected:Nn \__enumext_multicols_stop:
3344 {
3345     \int_compare:nNnT
3346     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3347     {
3348         \end{multicols}
3349         \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3350         {
3351             \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
3352         }
3353     }
3354 }

```

(End of definition for `\__enumext_multicols_stop:`)

`\__enumext_after_list:` The function `\__enumext_after_list:` will 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.

```

3355 \cs_new_protected:Nn \__enumext_after_list:
3356 {
3357     \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
3358     {
3359         \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3360         {
3361             \msg_warning:nn { enumext } { missing-miniright }
3362             \miniright
3363         }
3364         \int_gzero:N \g__enumext_minipage_stat_int
3365         \end{\__enumext_mini_env*}
3366         \par\addvspace { \l__enumext_minipage_after_skip }
3367     }
3368     { \__enumext_multicols_stop: }

```

If the `check-ans` key is active, we set the boolean variable `\g__enumext_check_ans_show_bool` to true and copy the “*store name*” to the variable `\g__enumext_store_name_tl`.

```

3369     \__enumext_check_ans_key_hook:

```

Now apply the `{\code}` handled by the `after` key together with the *vertical space* handled by the `below` key if they are present, set `\l__enumext_standar_bool` to false and save the *current value* of the counter for `series`, `resume` and `resume*` keys.

```

3370     \__enumext_after_stop_list:
3371     \__enumext_vspace_below:
3372     \bool_set_false:N \l__enumext_standar_bool
3373     \__enumext_resume_save_counter:
3374 }

```

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

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`.

```
3375 \__enumext_after_env:nn {enumext} { \__enumext_execute_after_env: }
```

### 11.36 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.

```
3376 \NewDocumentEnvironment{keyans}{0}{ }
3377 {
3378   \__enumext_keyans_safe_exec:
3379   \__enumext_keyans_parse_keys:n {#1}
3380   \__enumext_before_list_v:
3381   \__enumext_start_list:nn
3382   { \tl_use:N \l__enumext_label_v_tl }
3383   {
3384     \__enumext_list_arg_two_v:
3385     \__enumext_before_keys_exec_v:
3386   }
3387   \__enumext_after_args_exec_v:
3388 }
3389 {
3390   \__enumext_check_starred_cmd:n { item }
3391   \__enumext_stop_list:
3392   \__enumext_after_list_v:
3393 }
```

(End of definition for `keyans`. This function is documented on page 13.)

`\__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.

```
3394 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3395 {
3396   \bool_if:NF \l__enumext_store_active_bool
3397   {
3398     \msg_error:nnnn { enumext } { wrong-place } { keyans } { save-ans }
3399   }
3400   \int_incr:N \l__enumext_keyans_level_int
3401   \bool_set_true:N \l__enumext_keyans_env_bool
3402   \__enumext_keyans_start_line:
3403   % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3404   \bool_set_false:N \l__enumext_store_active_bool
3405   \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
3406   {
3407     \msg_error:nn { enumext } { keyans-nested }
3408   }
3409   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3410   {
3411     \msg_error:nn { enumext } { keyans-wrong-level }
3412   }
3413 }
```

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

`\__enumext_keyans_parse_keys:n` Parse [`<key = val>`] for `keyans` environment.

```
3414 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3415 {
3416   \keys_set:nn { enumext / keyans } {#1}
3417 }
```

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

`\__enumext_before_list_v:` The function `\__enumext_before_list_v:` will add the *vertical spacing above* the environment if the *above* key is active next to the *(code)* defined by the *before* key if it is active.

```

3418 \cs_new_protected:Nn \__enumext_before_list_v:
3419 {
3420   \__enumext_vspace_above_v:
3421   \__enumext_before_args_exec_v:

```

When the *mini-env* key is active it will set the value of the `\l__enumext_minipage_right_v_dim` to be the *width* of the `\__enumext_mini_env*` environment on the *left side*, using this value together with the value of the `\l__enumext_minipage_hsep_v_dim` set by the *mini-sep* key, the value of `\l__enumext_minipage_left_v_dim` will be set, which will be the *width* of `\__enumextt_mini_env*` environment on the *right side*, always having `\linewidth` as the maximum width between them.

```

3422   \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3423   {
3424     \dim_set:Nn \l__enumext_minipage_left_v_dim
3425     {
3426       \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3427     }

```

The boolean variable `\l__enumext_minipage_active_v_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_keyans_mini_addvspace:` is called and the `\__enumext_mini_env*` environment on *left side* will be initialized followed by the *vertical spacing* `\l__enumext_minipage_left_skip`. Here we use the plain TeX macro `\nointerlineskip` to prevent baseline “glue” being added between the next pair of boxes in a *vertical list*.

```

3428   \bool_set_true:N \l__enumext_minipage_active_v_bool
3429   \int_gincr:N \g__enumext_minipage_stat_int
3430   \__enumext_keyans_mini_addvspace:
3431   \nointerlineskip\noindent
3432   \begin{\__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
3433   }

```

After these actions, the `\__enumext_keyans_multicols_start:` function is called to handle the *multicols* environment.

```

3434   \__enumext_keyans_multicols_start:
3435   }

```

(End of definition for `\__enumext_before_list_v:`)

`\__enumext_keyans_multicols_start:` The function `\__enumext_keyans_multicols_start:` will start the *multicols* environment according to the value passed by the *columns* key.

```

3436 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3437 {
3438   \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3439   {

```

Set the default value for `\columnsep` when *columns-sep* key is *opt*.

```

3440     \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3441     {
3442       \dim_set:Nn \l__enumext_columns_sep_v_dim
3443       {
3444         (
3445           \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3446         ) / \l__enumext_columns_v_int
3447         - \l__enumext_listoffset_v_dim
3448       }
3449     }
3450     \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim

```

Then we will set the value of `\multicolsep` and `\columnseprule` equal to zero (we do not want a vertical rule in this environment).

```

3451     \skip_zero:N \multicolsep
3452     \dim_zero:N \columnseprule

```

We will calculate the *vertical spacing* settings for the *multicols* environment using the function `\__enumext_keyans_multi_addvspace:` and apply our “*vertical adjust spacing*”, then start the *multicols* environment.

```

3453     \bool_if:NF \l__enumext_minipage_active_v_bool
3454     {
3455       \__enumext_keyans_multi_addvspace:
3456     }
3457     \raggedcolumns

```

```

3458     \begin{multicols}{\l__enumext_columns_v_int }
3459   }
3460 }

```

(End of definition for `\__enumext_keyans_multicols_start:`)

`\__enumext_keyans_multicols_stop:`

The function `\__enumext_keyans_multicols_stop:` will stop the `multicols` environment. If the boolean variable `\l__enumext_minipage_active_v_bool` is false (not nested in `\__enumext_mini-env*`) we will apply our vertical “adjust” spacing.

```

3461 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3462 {
3463   \int_compare:nNtT { \l__enumext_columns_v_int } > { 1 }
3464   {
3465     \end{multicols}
3466     \bool_if:NF \l__enumext_minipage_active_v_bool
3467     {
3468       \par\addvspace{ \l__enumext_multicols_below_v_skip }
3469     }
3470   }
3471 }

```

(End of definition for `\__enumext_keyans_multicols_stop:`)

`\__enumext_after_list_v:`

The function `\__enumext_after_list_v:` will check the state of the boolean variable `\l__enumext_minipage_active_v_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 vertical adjustment space `\l__enumext_minipage_after_skip`, otherwise we will close the `multicols` environment.

```

3472 \cs_new_protected:Nn \__enumext_after_list_v:
3473 {
3474   \bool_if:NTF \l__enumext_minipage_active_v_bool
3475   {
3476     \int_compare:nNtT { \g__enumext_minipage_stat_int } = { 1 }
3477     {
3478       \msg_warning:nn { enumext } { missing-miniright }
3479       \miniright
3480     }
3481     \int_gzero:N \g__enumext_minipage_stat_int
3482     \end{\__enumext_mini-env*}
3483     \par\addvspace{ \l__enumext_minipage_after_skip }
3484   }
3485   { \__enumext_keyans_multicols_stop: }

```

Finally we will apply the `{\code}` handled by the `after` key together with the *vertical space* handled by the `below` key if they are present.

```

3486   \bool_set_false:N \l__enumext_keyans_env_bool
3487   \__enumext_after_stop_list_v:
3488   \__enumext_vspace_below_v:
3489 }

```

(End of definition for `\__enumext_after_list_v:`)

## 11.37 The environment `keyanspic` and `\anspic`

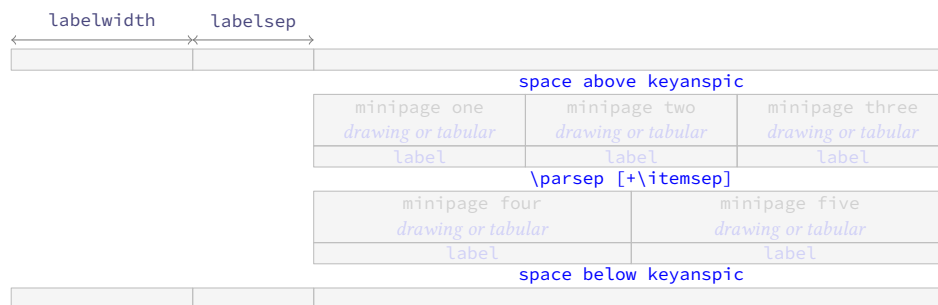
The `keyanspic` environment is a list-based environment that uses the same configuration for “spacing” and `\label` as the `keyans` environment, but it does not use `\item`.

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.

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?](#).



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

### 11.37.1 The command `\anspic`

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

```
3490 \NewDocumentCommand \anspic { s o +m }
3491 {
```

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

```
3492   \bool_if:NF \l__enumext_store_active_bool
3493   {
3494     \msg_error:nnnn { enumext } { wrong-place } { keyanspic } { save-ans }
3495   }
3496   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3497   {
3498     \msg_error:nn { enumext } { keyanspic-wrong-level }
3499   }
3500   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
3501   {
3502     \msg_error:nnnn { enumext } { command-wrong-place } { anspic } { keyans }
3503   }
```

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.

```
3504   \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
3505   {
3506     \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
3507   }
3508 }
```

(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 `\label`, which will have the same configuration as the `keyans` environment.

```
3509 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
3510 {
3511   \stepcounter { enumXvi }
3512   #3 \\\
3513   \bool_if:nT { #1 }
3514   {
3515     \__enumext_keyans_addto_prop:n { #2 }
3516     \__enumext_keyans_store_ref:
3517     \__enumext_keyans_addto_seq:n { #2 }
3518     \int_gincr:N \g__enumext_check_starred_cmd_int
3519     \bool_lazy_or:nnT
3520     { \bool_if_p:N \l__enumext_show_answer_bool }
3521     { \bool_if_p:N \l__enumext_show_position_bool }
3522     {
3523       \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
3524       \__enumext_keyans_show_left:n { #2 }
3525       \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
3526     }
3527   }
3528   \tl_use:N \l__enumext_label_font_style_v_tl
3529   \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
3530 }
```

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

### 11.37.2 The environment `keyanspic`

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

```

3531 \NewDocumentEnvironment{keyanspic}{ o }
3532 {
3533   \__enumext_keyans_pic_safe_exec:
3534   \__enumext_start_list:nn
3535   { }
3536   {
3537     \__enumext_keyans_pic_arg_two:
3538   }

```

We apply the “adjusted” vertical spacing above the environment

```

3539   \vspace { \__enumext_keyans_pic_above_skip }
3540 }

```

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.

```

3541 {
3542   \tl_if_novalue:nTF { #1 }
3543   {
3544     \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
3545   }
3546   { \__enumext_keyans_pic_do:n { #1 } }
3547   \__enumext_stop_list:
3548   \__enumext_check_starred_cmd:n { anspic }
3549   \setcounter { enumXvi } { 0 }
3550   \vspace { \__enumext_topsep_v_skip }
3551   %\bool_set_false:N \l__enumext_store_active_bool
3552 }

```

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

`\__enumext_keyans_pic_safe_exec:`

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

```

3553 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
3554 {
3555   \int_incr:N \l__enumext_keyans_pic_level_int
3556   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
3557   {
3558     \msg_error:nn { enumext } { keyanspic-nested }
3559   }
3560   \__enumext_keyans_start_line:
3561 }

```

(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`.

```

3562 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3563 {
3564   \dim_compare:nNnT { #1 } < { 0pt }
3565   { \skip_set:Nn #1 { -#1 } }
3566 }

```

(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.

```

3567 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
3568 {

```

The first thing to do is to set the boolean variable `\l__enumext_leftmargin_tmp_v_bool` handled by the `list-indent` key to false, then we copy the definition of the second list argument from the `keyans` environment.

```

3569   \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3570   \__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.

```

3571 \skip_add:Nn \parsep { \itemsep }
3572 \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
3573 \dim_zero:N \labelwidth
3574 \dim_zero:N \listparindent
3575 \dim_zero:N \labelsep
3576 \skip_zero:N \partopsep
3577 \skip_zero:N \itemsep

```

We set the value of `\l__enumxt_keyans_pic_above_skip` which we will use to apply our “adjust” space above `keyanspic`, finally we call `\__enumxt_item_std:w` followed by `\scan_stop:` to prevent the error message returned by  $\TeX$  when not using the `\item` command.

```

3578 \__enumxt_keyans_pic_skip_abs:N \parsep
3579 \skip_set:Nn \l__enumxt_keyans_pic_above_skip
3580 {
3581   \box_dp:N \strutbox
3582   + \l__enumxt_topsep_v_skip
3583   - \parsep
3584 }
3585 \__enumxt_item_std:w \scan_stop:
3586 }

```

(End of definition for `\__enumxt_keyans_pic_arg_two:`)

```

\__enumxt_keyans_pic_do:n
\__enumxt_keyans_pic_do:e

```

The optional argument is split by comma and is handled directly by the function `\__enumxt_keyans_pic_do:n` and passed to the function `\__enumxt_keyans_pic_row:n`.

```

3587 \cs_new_protected:Nn \__enumxt_keyans_pic_do:n
3588 {
3589   \clist_map_function:nN { #1 } \__enumxt_keyans_pic_row:n
3590 }
3591 \cs_generate_variant:Nn \__enumxt_keyans_pic_do:n { e }

```

(End of definition for `\__enumxt_keyans_pic_do:n`)

```
\__enumxt_keyans_pic_row:n
```

The function `\__enumxt_keyans_pic_row:n` will set the widths for the `minipage` environments and place the content  $\langle stored \rangle$  by `\anspic*` in the `\l__enumxt_keyans_pic_body_seq` sequence inside them.

```

3592 \cs_new_protected:Nn \__enumxt_keyans_pic_row:n
3593 {
3594   \dim_set:Nn \l__enumxt_keyans_pic_width_dim { \linewidth / #1 }
3595   \int_set:Nn \l__enumxt_keyans_pic_above_int { \l__enumxt_keyans_pic_below_int }
3596   \int_set:Nn \l__enumxt_keyans_pic_below_int { \l__enumxt_keyans_pic_above_int + #1 }
3597   \int_step_inline:nnn
3598     { \l__enumxt_keyans_pic_above_int + 1 }
3599     { \l__enumxt_keyans_pic_below_int }
3600     {
3601       \__enumxt_minipage:w [ b ] { \l__enumxt_keyans_pic_width_dim }
3602       \centering
3603       \seq_item:Nn \l__enumxt_keyans_pic_body_seq { ##1 }
3604       \__enumxt_endminipage:
3605     }
3606   \par
3607 }

```

(End of definition for `\__enumxt_keyans_pic_row:n`)

## 11.38 The horizontal environments

Generating horizontal list environments is NOT as simple as standard  $\TeX$  list environments. The fundamental part of the code is adapted from the `shortlst` package to a more modern version using `expl3`. It is not possible to redefine `\item` and `\makelabel` as in the non starred versions (at least I have not achieved it) and as we will make it behave differently, we have no other option than to define a cascade of functions.

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 ( $\langle number \rangle$ ).

### 11.38.1 Functions for item box width

\\_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 and create `\itemwidth` in a public form.

```

3608 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
3609 {
3610   \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3611   {
3612     \dim_set:Nn \l__enumext_columns_sep_vii_dim
3613     {
3614       ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
3615       / \l__enumext_columns_vii_int
3616     }
3617   }
3618   \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - 1 }
3619   \dim_set:Nn \l__enumext_item_width_vii_dim
3620   {
3621     ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
3622     / \l__enumext_columns_vii_int - \l__enumext_labelwidth_vii_dim
3623     - \l__enumext_labelsep_vii_dim
3624   }
3625   \dim_zero_new:N \itemwidth
3626 }
3627 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3628 {
3629   \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3630   {
3631     \dim_set:Nn \l__enumext_columns_sep_viii_dim
3632     {
3633       ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
3634       / \l__enumext_columns_viii_int
3635     }
3636   }
3637   \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - 1 }
3638   \dim_set:Nn \l__enumext_item_width_viii_dim
3639   {
3640     ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
3641     / \l__enumext_columns_viii_int - \l__enumext_labelwidth_viii_dim
3642     - \l__enumext_labelsep_viii_dim
3643   }
3644   \dim_zero_new:N \itemwidth
3645 }
```

(End of definition for \\_enumext\_starred\_columns\_set\_vii: and \\_enumext\_starred\_columns\_set\_viii:.)

### 11.38.2 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`.

```

3646 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
3647 {
3648   \int_set:Nn \l__enumext_joined_item_vii_int {#1}
3649   \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
3650   {
3651     \msg_warning:nnee { enumext } { item-joined }
3652     { \int_use:N \l__enumext_joined_item_vii_int }
3653     { \int_use:N \l__enumext_columns_vii_int }
3654     \int_set:Nn \l__enumext_joined_item_vii_int
3655     {
3656       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
3657     }
3658   }
3659   \int_compare:nNnT
3660   { \l__enumext_joined_item_vii_int }
3661   >
3662   { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
3663   {
3664     \msg_warning:nnee { enumext } { item-joined-columns }
3665     { \int_use:N \l__enumext_joined_item_vii_int }
3666     {
3667       \int_eval:n
```

```

3668         { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
3669     }
3670     \int_set:Nn \l__enumext_joined_item_vii_int
3671     {
3672         \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
3673     }
3674 }
3675 \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { 1 }
3676 {
3677     \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
3678     \int_decr:N \l__enumext_joined_item_aux_vii_int
3679     \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3680     \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3681     \dim_set:Nn \l__enumext_joined_width_vii_dim
3682     {
3683         \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
3684         + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3685             + \l__enumext_columns_sep_vii_dim
3686             )*\l__enumext_joined_item_aux_vii_int
3687     }
3688     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
3689 }
3690 {
3691     \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
3692     \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
3693 }
3694 }
3695 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
3696 {
3697     \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3698     \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3699     {
3700         \msg_warning:nnee { enumext } { item-joined }
3701         { \int_use:N \l__enumext_joined_item_viii_int }
3702         { \int_use:N \l__enumext_columns_viii_int }
3703         \int_set:Nn \l__enumext_joined_item_viii_int
3704         {
3705             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
3706         }
3707     }
3708     \int_compare:nNnT
3709     { \l__enumext_joined_item_viii_int }
3710     >
3711     { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
3712     {
3713         \msg_warning:nnee { enumext } { item-joined-columns }
3714         { \int_use:N \l__enumext_joined_item_viii_int }
3715         {
3716             \int_eval:n
3717             { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
3718         }
3719         \int_set:Nn \l__enumext_joined_item_viii_int
3720         {
3721             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
3722         }
3723     }
3724     \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { 1 }
3725     {
3726         \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3727         \int_decr:N \l__enumext_joined_item_aux_viii_int
3728         \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
3729         \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
3730         \dim_set:Nn \l__enumext_joined_width_viii_dim
3731         {
3732             \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3733             + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3734                 + \l__enumext_columns_sep_viii_dim
3735                 )*\l__enumext_joined_item_aux_viii_int
3736         }
3737         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3738     }

```

```

3739     {
3740         \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3741         \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3742     }
3743 }

```

(End of definition for \\_\_enumext\_starred\_joined\_item\_vii:n and \\_\_enumext\_starred\_joined\_item\_viii:n)

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

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

```

3744 \cs_new_protected:Nn \__enumext_start_mini_vii:
3745 {
3746     \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
3747     {
3748         \dim_set:Nn \l__enumext_minipage_left_vii_dim
3749         {
3750             \linewidth
3751             - \l__enumext_minipage_right_vii_dim
3752             - \l__enumext_minipage_hsep_vii_dim
3753         }
3754         \bool_set_true:N \l__enumext_minipage_active_vii_bool
3755         \dim_gset_eq:NN
3756         \g__enumext_minipage_right_vii_dim
3757         \l__enumext_minipage_right_vii_dim
3758         \__enumext_mini_addvspace_vii:
3759         \nointerlineskip\noindent
3760         \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
3761     }
3762 }

```

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

```

3763 \cs_new_protected:Nn \__enumext_stop_mini_vii:
3764 {
3765     \bool_if:NT \l__enumext_minipage_active_vii_bool
3766     {
3767         \end{__enumext_mini_env*}
3768         \hfill
3769         \bool_gset_true:N \g__enumext_minipage_active_vii_bool
3770     }
3771 }

```

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.

```

3772 \__enumext_after_env:nn {enumext*}
3773 {
3774     \bool_if:NT \g__enumext_minipage_active_vii_bool
3775     {
3776         \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
3777         \par\addvspace { \g__enumext_minipage_right_skip }
3778         \bool_if:NF \g__enumext_minipage_center_vii_bool
3779         {
3780             \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
3781             {
3782                 \centering
3783             }
3784         }
3785         \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
3786         {
3787             \tl_use:N \g__enumext_miniright_code_vii_tl
3788         }
3789         \box_use_drop:N \l__enumext_miniright_code_vii_box
3790         \end{__enumext_mini_env*}
3791         \par\addvspace{ \g__enumext_minipage_after_skip }

```

```

3792     }
3793     \bool_gset_false:N \g__enumext_minipage_active_vii_bool
3794     \bool_gset_true:N \g__enumext_minipage_center_vii_bool
3795     \tl_gclear:N \g__enumext_miniright_code_vii_tl
3796     \dim_gzero:N \g__enumext_minipage_right_vii_dim
3797     \bool_gset_false:N \g__enumext_starred_bool
3798 }

```

(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.

```

3799 \cs_new_protected:Nn \__enumext_start_mini_viii:
3800 {
3801     \dim_compare:nNtT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3802     {
3803         \dim_set:Nn \l__enumext_minipage_left_viii_dim
3804         {
3805             \linewidth
3806             - \l__enumext_minipage_right_viii_dim
3807             - \l__enumext_minipage_hsep_viii_dim
3808         }
3809         \bool_set_true:N \l__enumext_minipage_active_viii_bool
3810         \dim_gset_eq:NN
3811             \g__enumext_minipage_right_viii_dim
3812             \l__enumext_minipage_right_viii_dim
3813         \__enumext_mini_addvspace_viii:
3814         \nointerlineskip\noindent
3815         \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
3816     }
3817 }
3818 \cs_new_protected:Nn \__enumext_stop_mini_viii:
3819 {
3820     \bool_if:NT \l__enumext_minipage_active_viii_bool
3821     {
3822         \end{__enumext_mini_env*}
3823         \hfill
3824         \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3825     }
3826 }
3827 \__enumext_after_env:nn {keyans*}
3828 {
3829     \bool_if:NT \g__enumext_minipage_active_viii_bool
3830     {
3831         \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3832         \par\addvspace { \g__enumext_minipage_right_skip }
3833         \bool_if:NF \g__enumext_minipage_center_viii_bool
3834         {
3835             \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
3836             {
3837                 \centering
3838             }
3839         }
3840         \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
3841         {
3842             \tl_use:N \g__enumext_miniright_code_viii_tl
3843         }
3844         \box_use_drop:N \l__enumext_miniright_code_viii_box
3845         \end{__enumext_mini_env*}
3846         \par\addvspace{ \g__enumext_minipage_after_skip }
3847     }
3848     \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3849     \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3850     \tl_gclear:N \g__enumext_miniright_code_viii_tl
3851     \dim_gzero:N \g__enumext_minipage_right_viii_dim
3852 }

```

(End of definition for `\__enumext_start_mini_viii:` and `\__enumext_stop_mini_viii:`)



### 11.39 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.

```

3853 \NewDocumentEnvironment{enumext*}{o}{
3854 {
3855   \__enumext_safe_exec_vii:
3856   \__enumext_parse_keys_vii:n {#1}
3857   \__enumext_starred_columns_set_vii:
3858   \__enumext_before_list_vii:
3859   \__enumext_start_store_level_vii:
3860   \__enumext_start_list:nn { }
3861   {
3862     \__enumext_list_arg_two_vii:
3863     \__enumext_before_keys_exec_vii:
3864   }
3865   \item[] \scan_stop:
3866   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
3867   \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
3868 }
3869 {
3870   \__enumext_stop_item_tmp_vii:
3871   \__enumext_remove_extra_parsep_vii:
3872   \__enumext_stop_list:
3873   \__enumext_stop_store_level_vii:
3874   \__enumext_after_list_vii:
3875 }
```

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

`\__enumext_safe_exec_vii:` First check the maximum nesting level for the `enumext*` environment then set the vars `\l__enumext_starred_bool` and `\g__enumext_starred_bool`.

```

3876 \cs_new_protected:Nn \__enumext_safe_exec_vii:
3877 {
3878   \__enumext_internal_mini_page:
3879   \__enumext_is_not_nested:
3880   \int_incr:N \l__enumext_level_h_int
3881   \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
3882   {
3883     \msg_error:nn { enumext } { nested }
3884   }
3885   \bool_set_true:N \l__enumext_starred_bool
3886   \__enumext_is_on_first_level:
3887 }
```

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

`\__enumext_parse_keys_vii:n` Parse `[⟨key = val⟩]` for `enumext*`. If the variable `\l__enumext_store_active_bool` is true it will call the functions `\__enumext_parse_series:n` and `\__enumext_store_active_keys_vii:n` and reprocess the `⟨keys⟩` to pass them to the storage `⟨sequence⟩`.

```

3888 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
3889 {
3890   \tl_if_novalue:nF {#1}
3891   {
3892     \str_clear:N \l__enumext_series_str
3893     \keys_set:nn { enumext / enumext* } {#1}
3894     \__enumext_parse_series:n {#1}
3895     \__enumext_store_active_keys_vii:n {#1}
3896     \__enumext_nested_base_line_fix:
3897   }
3898 }
```

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

`\__enumext_before_list_vii:` The function `\__enumext_before_list_vii:` 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_vii:` handle by `mini-env`.

```

3899 \cs_new_protected:Nn \__enumext_before_list_vii:
3900 {
3901   \__enumext_vspace_above_vii:
```

```

3902     \__enumext_check_ans_active:
3903     \__enumext_before_args_exec_vii:
3904     \__enumext_start_mini_vii:
3905 }

```

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

\\_\_enumext\_after\_list\_vii:

The function \\_\_enumext\_after\_list: first call the function \\_\_enumext\_stop\_mini\_vii:, then apply the `{\code}` handled by the `after` key together with the *vertical space* handled by the `below` key if they are present. Finally set false the vars \g\_\_enumext\_starred\_bool and \l\_\_enumext\_starred\_bool, save the *current value* of the counter in \g\_\_enumext\_resume\_vii\_int for the `resume` key. If the `save-ans` key is active, it will create the integer variable for the `resume` key, we only have to assign it the value of the current counter.

```

3906 \cs_new_protected:Nn \__enumext_after_list_vii:
3907 {
3908     \__enumext_stop_mini_vii:
3909     \__enumext_after_stop_list_vii:
3910     \__enumext_check_ans_key_hook:
3911     \__enumext_vspace_below_vii:
3912     \bool_set_false:N \l__enumext_starred_bool
3913     \__enumext_resume_save_counter:
3914 }

```

(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 if `enumext*` are nested in `enumext`.

```

3915 \cs_new_protected:Nn \__enumext_start_store_level_vii:
3916 {
3917     \bool_if:NT \l__enumext_store_active_bool
3918     {
3919         \int_compare:nNt { \l__enumext_level_int } > { 0 }
3920         {
3921             \__enumext_store_level_open_vii:
3922         }
3923     }
3924 }
3925 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
3926 {
3927     \bool_if:NT \l__enumext_store_active_bool
3928     {
3929         \int_compare:nNt { \l__enumext_level_int } > { 0 }
3930         {
3931             \__enumext_store_level_close_vii:
3932         }
3933     }
3934 }

```

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

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

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

First we will call the function \\_\_enumext\_stop\_item\_tmp\_vii: that we will redefine later, we will increment the value of \l\_\_enumext\_item\_column\_pos\_vii\_int that will count the item's by rows and the value of \g\_\_enumext\_item\_count\_all\_vii\_int that will count the total of item's in the environment. After that we will call the function \\_\_enumext\_item\_peek\_args\_vii: that will handle the arguments passed to `\item`.

```

3935 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
3936 {
3937     \__enumext_stop_item_tmp_vii:
3938     \int_incr:N \l__enumext_item_column_pos_vii_int
3939     \int_gincr:N \g__enumext_item_count_all_vii_int
3940     \__enumext_item_peek_args_vii:
3941 }

```

(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).

```

3942 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
3943 {
3944   \peek_meaning:NTF (
3945     { \__enumext_joined_item_vii:w }
3946     { \__enumext_joined_item_vii:w (1) }
3947   }

```

(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`.

```

3948 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
3949 {
3950   \__enumext_starred_joined_item_vii:n {#1}
3951   \peek_meaning_remove:NTF *
3952     { \__enumext_starred_item_vii:w }
3953     { \__enumext_standar_item_vii:w }
3954 }

```

(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]`.

```

3955 \cs_new_protected:Npn \__enumext_standar_item_vii:w
3956 {
3957   \bool_set_false:N \l__enumext_item_starred_vii_bool
3958   \peek_meaning:NTF [
3959     {
3960       \bool_set_eq:NN
3961         \l__enumext_wrap_label_vii_bool
3962         \l__enumext_wrap_label_opt_vii_bool
3963       \__enumext_start_item_vii:w
3964     }
3965     {
3966       \bool_set_true:N \l__enumext_wrap_label_vii_bool
3967       \legacy_if_set_true:n { @noitemarg }
3968       \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3969     }
3970   }

```

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

`\__enumext_starred_item_vii:w`  
`\__enumext_starred_item_vii_aux_i:w`  
`\__enumext_starred_item_vii_aux_ii:w`  
`\__enumext_starred_item_vii_aux_iii: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>]`.

```

3971 \cs_new_protected:Npn \__enumext_starred_item_vii:w
3972 {
3973   \bool_set_true:N \l__enumext_item_starred_vii_bool
3974   \bool_set_true:N \l__enumext_wrap_label_vii_bool
3975   \peek_meaning:NTF [
3976     { \__enumext_starred_item_vii_aux_i:w }
3977     { \__enumext_starred_item_vii_aux_ii:w }
3978   }
3979   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
3980   {
3981     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
3982     \__enumext_starred_item_vii_aux_ii:w
3983   }
3984   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w

```

```

3985 {
3986   \peek_meaning:NTF [
3987     { \__enumext_starred_item_vii_aux_iii:w }
3988     {
3989       \dim_set_eq:NN
3990       \l__enumext_item_symbol_sep_vii_dim
3991       \l__enumext_labelsep_vii_dim
3992       \legacy_if_set_true:n { @noitemarg }
3993       \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
3994     }
3995   }
3996   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
3997   {
3998     \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
3999     \legacy_if_set_true:n { @noitemarg }
4000     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4001   }

```

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

### 11.39.2 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.

```

4002 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
4003 {
4004   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
4005   \legacy_if:nT { @noitemarg }
4006   {
4007     \legacy_if_set_false:n { @noitemarg }
4008     \legacy_if:nT { @nmbrrlist }
4009     {
4010       \bool_if:NT \l__enumext_hyperref_bool
4011       {
4012         \legacy_if_set_true:n { @hyper@item }
4013       }
4014       \refstepcounter{enumXvii}
4015       \bool_if:NT \l__enumext_check_answers_bool
4016       {
4017         \int_gincr:N \g__enumext_item_number_int
4018         \bool_set_true:N \l__enumext_item_number_bool
4019       }
4020     }
4021   }

```

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

```

4022 \group_begin:
4023   \lrbox{ \l__enumext_item_text_vii_box }
4024   \bool_if:NF \l__enumext_footnotes_key_bool
4025   {
4026     \__enumext_renew_footnote:
4027   }
4028   \bool_if:NT \l__enumext_item_starred_vii_bool
4029   {
4030     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
4031     {
4032       \tl_gset_eq:NN
4033       \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
4034     }
4035     \mode_leave_vertical:

```

```

4036         \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
4037         \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
4038         \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
4039         \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
4040     }
4041     \group_begin:
4042     \tl_use:N \l__enumext_label_font_style_vii_tl
4043     \bool_if:NTF \l__enumext_wrap_label_vii_bool
4044     {
4045         \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
4046         { \__enumext_wrapper_label_vii:n {#1} }
4047     }
4048     {
4049         \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
4050     }
4051     \group_end:
4052     \skip_horizontal:N \l__enumext_labelsep_vii_dim
4053     \tl_use:N \l__enumext_after_list_args_vii_tl
4054     \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
4055     \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
4056     \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
4057     \tl_use:N \l__enumext_fake_item_indent_vii_tl
4058 }

```

(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.

```

4059 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
4060 {
4061     \__enumext_endminipage:
4062     \endlrbox
4063     \group_end:
4064     \box_set_wd:Nn \l__enumext_item_text_vii_box
4065     {
4066         \l__enumext_joined_width_vii_dim
4067         + \l__enumext_labelwidth_vii_dim
4068         + \l__enumext_labelsep_vii_dim
4069     }
4070     \int_set:Nn \hbadness { 10000 }
4071     \box_use_drop:N \l__enumext_item_text_vii_box
4072     \bool_if:NF \l__enumext_footnotes_key_bool
4073     {
4074         \__enumext_print_footnote:
4075     }
4076     \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
4077     {
4078         \par\noindent
4079         \int_zero:N \l__enumext_item_column_pos_vii_int
4080     }
4081     { \hspace{ \l__enumext_columns_sep_vii_dim } }
4082 }

```

(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.

```

4083 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
4084 {
4085     \int_compare:nNnTF
4086     {
4087         \int_mod:nn { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
4088     }
4089     =
4090     { 0 }
4091     {
4092         \par
4093         \vspace{ -\l__enumext_itemsep_vii_skip }
4094         \int_gzero:N \g__enumext_item_count_all_vii_int
4095     }
4096 }

```

(End of definition for `\__enumext_remove_extra_parsep_viii:`)

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`.

```
4097 \__enumext_after_env:nn {enumext*} { \__enumext_execute_after_env: }
```

## 11.40 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.

```
4098 \NewDocumentEnvironment{keyans*}{o }
4099 {
4100   \__enumext_safe_exec_viii:
4101   \__enumext_parse_keys_viii:n {#1}
4102   \__enumext_starred_columns_set_viii:
4103   \__enumext_before_list_viii:
4104   \__enumext_start_list:nn { }
4105   {
4106     \__enumext_list_arg_two_viii:
4107     \__enumext_before_keys_exec_viii:
4108   }
4109   \item[] \scan_stop:
4110   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
4111   \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
4112 }
4113 {
4114   \__enumext_stop_item_tmp_viii:
4115   \__enumext_remove_extra_parsep_viii:
4116   \__enumext_check_starred_cmd:n { item }
4117   \__enumext_stop_list:
4118   \__enumext_after_list_viii:
4119 }
```

(End of definition for `keyans*`. This function is documented on page 13.)

`\__enumext_safe_exec_viii:`

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

```
4120 \cs_new_protected:Nn \__enumext_safe_exec_viii:
4121 {
4122   \int_incr:N \__enumext_keyans_level_h_int
4123   \int_compare:nNt { \__enumext_keyans_level_h_int } > { 1 }
4124   {
4125     \msg_error:nn { enumext } { nested }
4126   }
4127   \__enumext_keyans_start_line:
4128   % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
4129   \bool_set_false:N \__enumext_store_active_bool
4130   \int_compare:nNt { \__enumext_level_int } > { 1 }
4131   {
4132     \msg_error:nn { enumext } { keyans-wrong-level }
4133   }
4134 }
```

(End of definition for `\__enumext_safe_exec_viii:`)

`\__enumext_parse_keys_viii:n`

Parse [`<key = val>`] for `keyans*`.

```
4135 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
4136 {
4137   \tl_if_novalue:nF {#1}
4138   {
4139     \keys_set:nn { enumext / keyans* } {#1}
4140   }
4141 }
```

(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`.

```
4142 \cs_new_protected:Nn \__enumext_before_list_viii:
4143 {
4144   \__enumext_vspace_above_viii:
```

```

4145     \__enumext_before_args_exec_viii:
4146     \__enumext_start_mini_viii:
4147 }

```

(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.

```

4148 \cs_new_protected:Nn \__enumext_after_list_viii:
4149 {
4150     \__enumext_stop_mini_viii:
4151     \__enumext_after_stop_list_viii:
4152     \__enumext_vspace_below_viii:
4153 }

```

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

#### 11.40.1 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 \\_\_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`.

```

4154 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
4155 {
4156     \__enumext_stop_item_tmp_viii:
4157     \int_incr:N \__enumext_item_column_pos_viii_int
4158     \int_gincr:N \g__enumext_item_count_all_viii_int
4159     \__enumext_item_peek_args_viii:
4160 }

```

(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)`.

```

4161 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
4162 {
4163     \peek_meaning:NTF (
4164     { \__enumext_joined_item_viii:w }
4165     { \__enumext_joined_item_viii:w (1) }
4166 }

```

(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.

```

4167 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
4168 {
4169     \__enumext_starred_joined_item_viii:n {#1}
4170     \peek_meaning_remove:NTF *
4171     { \__enumext_starred_item_viii:w }
4172     { \__enumext_standar_item_viii:w }
4173 }

```

(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 \\enumext\_wrap\_label\_opt\_viii\_bool equal to the state of the variable \\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 \\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 [ \\enumext\_label\_viii\_tl ].

```
4174 \\cs_new_protected:Npn \\enumext_standar_item_viii:w
4175 {
4176   \\bool_set_false:N \\enumext_item_starred_viii_bool
4177   \\peek_meaning:NTF [
4178     {
4179       \\bool_set_eq:NN
4180       \\enumext_wrap_label_viii_bool
4181       \\enumext_wrap_label_opt_viii_bool
4182       \\enumext_start_item_viii:w
4183     }
4184     {
4185       \\bool_set_true:N \\enumext_wrap_label_viii_bool
4186       \\legacy_if_set_true:n { @noitemarg }
4187       \\enumext_start_item_viii:w [ \\enumext_label_viii_tl ]
4188     }
4189   }
```

(End of definition for \\enumext\_standar\_item\_viii:w.)

\\enumext\_starred\_item\_viii: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\*[*content*].

\\enumext\_starred\_item\_viii\_aux\_i:w

\\enumext\_starred\_item\_viii\_aux\_ii:w

```
4190 \\cs_new_protected:Npn \\enumext_starred_item_viii:w
4191 {
4192   \\bool_set_true:N \\enumext_item_starred_viii_bool
4193   \\bool_set_true:N \\enumext_wrap_label_viii_bool
4194   \\peek_meaning:NTF [
4195     { \\enumext_starred_item_viii_aux_i:w }
4196     { \\enumext_starred_item_viii_aux_ii:w }
4197   }
```

The function \\enumext\_starred\_item\_viii\_aux\_i:w will save the optional argument to \\item\* in \\enumext\_store\_current\_opt\_arg\_tl and will save this argument along with the spacing set by the key `save-sep` in variable \\enumext\_store\_current\_label\_tl if present, then call the function \\enumext\_starred\_item\_viii\_aux\_ii:w.

```
4198 \\cs_new_protected:Npn \\enumext_starred_item_viii_aux_i:w [#1]
4199 {
4200   \\tl_clear:N \\enumext_store_current_label_tl
4201   \\tl_if_no_value:nF { #1 }
4202   {
4203     \\tl_if_empty:NF \\enumext_store_keyans_item_opt_sep_tl
4204     {
4205       \\tl_put_right:Ne \\enumext_store_current_label_tl { \\enumext_store_keyans_item_opt_sep_tl }
4206       \\tl_put_right:Ne \\enumext_store_current_label_tl { #1 }
4207     }
4208     \\tl_set:Ne \\enumext_store_current_opt_arg_tl { #1 }
4209   }
4210   \\enumext_starred_item_viii_aux_ii:w
4211 }
4212 \\cs_new_protected:Npn \\enumext_starred_item_viii_aux_ii:w
4213 {
4214   \\legacy_if_set_true:n { @noitemarg }
4215   \\enumext_start_item_viii:w [ \\enumext_label_viii_tl ]
4216 }
```

(End of definition for \\enumext\_starred\_item\_viii:w, \\enumext\_starred\_item\_viii\_aux\_i:w, and \\enumext\_starred\_item\_viii\_aux\_ii:w.)

\\enumext\_starred\_item\_exec:

The function \\enumext\_starred\_item\_exec: will be in charge of storing the current *label* for \\item\* followed by the [ *content* ] for \\item\*[ *content* ] if present in the *sequence* and *prop list* set by the `save-ans` key. In this same function the keys `show-ans`, `show-pos` and `save-ref` are implemented.

```
4217 \\cs_new_protected:Nn \\enumext_starred_item_exec:
```

```

4218 {
4219   \tl_put_left:Ne \l__enumext_store_current_label_tl { \l__enumext_label_viii_tl }
4220   \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
4221   \__enumext_keyans_store_ref:
4222   \tl_put_left:Ne \l__enumext_store_current_label_tl { \item }
4223   \__enumext_keyans_addto_seq_link:
4224   \int_gincr:N \g__enumext_check_starred_cmd_int
4225   \bool_if:NT \l__enumext_show_answer_bool
4226   {
4227     \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4228   }
4229   \bool_if:NT \l__enumext_show_position_bool
4230   {
4231     \tl_set:Ne \l__enumext_mark_answer_sym_tl
4232     {
4233       \group_begin:
4234       \exp_not:N \normalfont
4235       \exp_not:N \footnotesize [ \int_eval:n
4236       {
4237         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
4238       }
4239       ]
4240       \group_end:
4241     }
4242     \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4243   }
4244 }

```

(End of definition for `\__enumext_starred_item_exec:`)

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

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

```

4245 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
4246 {
4247   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
4248   \legacy_if:nT { @noitemarg }
4249   {
4250     \legacy_if_set_false:n { @noitemarg }
4251     \legacy_if:nT { @nmbrrlist }
4252     {
4253       \bool_if:NT \l__enumext_hyperref_bool
4254       {
4255         \legacy_if_set_true:n { @hyper@item }
4256       }
4257       \refstepcounter{enumXviii}
4258     }
4259   }

```

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

```

4260   \group_begin:
4261   \lrbox{ \l__enumext_item_text_viii_box }
4262   \bool_if:NF \l__enumext_footnotes_key_bool
4263   {
4264     \__enumext_renew_footnote:
4265   }
4266   \bool_if:NT \l__enumext_item_starred_viii_bool
4267   {
4268     \__enumext_starred_item_exec:
4269   }
4270   \group_begin:
4271   \tl_use:N \l__enumext_label_font_style_viii_tl
4272   \bool_if:NTF \l__enumext_wrap_label_viii_bool
4273   {
4274     \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
4275     { \__enumext_wrapper_label_viii:n {#1} }
4276   }
4277   {
4278     \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1 }
4279   }
4280   \group_end:
4281   \skip_horizontal:N \l__enumext_labelsep_viii_dim

```

```

4282 \tl_use:N \l__enumext_after_list_args_viii_tl
4283 \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
4284 \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
4285 \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
4286 \bool_if:NT \l__enumext_item_starred_viii_bool
4287 {
4288   \tl_use:N \l__enumext_fake_item_indent_viii_tl
4289   \__enumext_keyans_show_item_opt:
4290   \skip_horizontal:n { -\l__enumext_fake_item_indent_viii_dim - \l__enumext_labelsep_viii_dim }
4291 }
4292 {
4293   \tl_use:N \l__enumext_fake_item_indent_viii_tl
4294 }
4295 }

```

(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.

```

4296 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
4297 {
4298   \__enumext_endminipage:
4299   \endlrbox
4300   \group_end:
4301   \box_set_wd:Nn \l__enumext_item_text_viii_box
4302   {
4303     \l__enumext_joined_width_viii_dim
4304     + \l__enumext_labelwidth_viii_dim
4305     + \l__enumext_labelsep_viii_dim
4306   }
4307   \int_set:Nn \hbadness { 10000 }
4308   \box_use_drop:N \l__enumext_item_text_viii_box
4309   \bool_if:NF \l__enumext_footnotes_key_bool
4310   {
4311     \__enumext_print_footnote:
4312   }
4313   \int_compare:nNnTF
4314   { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
4315   {
4316     \par\noindent
4317     \int_zero:N \l__enumext_item_column_pos_viii_int
4318   }
4319   { \hspace{ \l__enumext_columns_sep_viii_dim } }
4320 }

```

(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.

```

4321 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
4322 {
4323   \int_compare:nNnT
4324   {
4325     \int_mod:nn
4326     { \g__enumext_item_count_all_viii_int }
4327     { \l__enumext_columns_viii_int }
4328   }
4329   =
4330   { 0 }
4331   {
4332     \par
4333     \vspace{ -\l__enumext_itemsep_viii_skip }
4334     \int_gzero:N \g__enumext_item_count_all_viii_int
4335   }
4336 }

```

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

### 11.41 The command \getkeyans

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

```
4337 \NewDocumentCommand \getkeyans { m }
4338 {
4339   \exp_args:Ne \__enumext_getkeyans_aux:n
4340   { \tl_to_str:e { \text_expand:n {#1} } }
4341 }
```

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

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

```
4342 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
4343 {
4344   \str_if_in:nnTF {#1} { : }
4345   {
4346     \use:e
4347     {
4348       \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
4349       { {##1} {##2} }
4350     }
4351     \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
4352   }
4353   { \msg_error:nnn { enumext } { missing-colon } {#1} }
4354 }
```

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

`\__enumext_getkeyans:nn` The internal function `\__enumext_getkeyans:nn` will check for the existence of the  $\langle \text{prop list} \rangle$ , if it does not exist it will return an error message, then it will fetch the content specified by the second  $\langle \text{argument} \rangle$  from  $\langle \text{prop list} \rangle$ .

```
4355 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
4356 {
4357   \prop_if_exist:cF { g__enumext_#1_prop }
4358   { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
4359   \group_begin:
4360   \prop_item:cn { g__enumext_#1_prop }{#2}
4361   \group_end:
4362 }
```

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

### 11.42 The command \printkeyans

The `\printkeyans` command prints “all stored content” in the  $\langle \text{sequence} \rangle$  defined by the `save-ans` key. The first thing we will do is define a set of  $\langle \text{filtered keys} \rangle$  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  $\langle \text{keys} \rangle$  for `\printkeyans*` and will be set by `\setenumext[ $\langle \text{print}^* \rangle$ ]` and the variable `\l__enumext_print_keyans_vii_tl` will have the default keys for the environment `enumext*` nested within the  $\langle \text{sequence} \rangle$  and will be set by `\setenumext[ $\langle \text{print}^*, * \rangle$ ]`, the rest of the variables will be for the environment `enumext` and will be set by `\setenumext[ $\langle \text{print}, \text{level} \rangle$ ]`

```
4363 \cs_generate_variant:Nn \keys_precompile:nnN { neN }
4364 \keys_define:nn { enumext / print }
4365 {
4366   print* .code:n = \keys_precompile:neN { enumext / enumext* }
4367               { \__enumext_filter_save_key:n {#1} }
4368               \l__enumext_print_keyans_starred_tl, % starred cmd
4369   print* .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4370   print-1 .code:n = \keys_precompile:neN { enumext / level-1 }
4371               { \__enumext_filter_save_key:n {#1} }
4372               \l__enumext_print_keyans_ii_tl,
4373   print-1 .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4374   print-2 .code:n = \keys_precompile:neN { enumext / level-2 }
4375               { \__enumext_filter_save_key:n {#1} }
4376               \l__enumext_print_keyans_ii_tl,
4377   print-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
4378   print-3 .code:n = \keys_precompile:neN { enumext / level-3 }
```

```

4379         { \__enumext_filter_save_key:n {#1} }
4380         \__enumext_print_keyans_iii_tl,
4381     print-3 .initial:n = { nosep, label=\roman*., first=\small, font=\small },
4382     print-4 .code:n    = \keys_precompile:neN { enumext / level-4 }
4383         { \__enumext_filter_save_key:n {#1} }
4384         \__enumext_print_keyans_iv_tl,
4385     print-4 .initial:n = { nosep, label=\Alph*., first=\small, font=\small },
4386     print-* .code:n    = \keys_precompile:neN { enumext / enumext* }
4387         { \__enumext_filter_save_key:n {#1} }
4388         \__enumext_print_keyans_vii_tl, % starred nested
4389     print-* .initial:n = { nosep, label=\arabic*., first=\small, font=\small },
4390 }

```

• The reason for storing  $\langle keys \rangle$  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  $\langle sequence \rangle$  for `\anskey`, `anskey*`, `\item*` and `\anspic*`. Within a group we will run our “precompiled keys” and then call the internal function `\__enumext_printkeyans:nnn`.

```

4391 \NewDocumentCommand \printkeyans { s O{ } m }
4392 {
4393     \group_begin:
4394         \tl_use:N \__enumext_print_keyans_i_tl
4395         \tl_use:N \__enumext_print_keyans_ii_tl
4396         \tl_use:N \__enumext_print_keyans_iii_tl
4397         \tl_use:N \__enumext_print_keyans_iv_tl
4398         \tl_use:N \__enumext_print_keyans_vii_tl
4399         \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
4400     \group_end:
4401 }

```

(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  $\langle sequence \rangle$ , if it does not exist it will return an error message, then it will check if not empty.

```

4402 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
4403 {
4404     \seq_if_exist:cTF { g__enumext_#3_seq }
4405     {
4406         \seq_if_empty:cF { g__enumext_#3_seq }
4407         {
4408             %%\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  $\langle sequence \rangle$ , then execute the variable `\__enumext_print_keyans_starred_tl` that contains the default  $\langle keys \rangle$  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  $\langle sequence \rangle$ .

```

4409         \bool_if:nTF {#1}
4410         {
4411             \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
4412             {
4413                 \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
4414             }
4415             {
4416                 \tl_use:N \__enumext_print_keyans_starred_tl
4417                 \begin{enumext*}[#2]
4418                     \keys_set:nn { enumext / level-1 }{ base-fix }
4419                     \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4420                 \end{enumext*}
4421             }
4422         }

```

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

```

4423         {
4424             \begin{enumext}[#2]
4425                 \keys_set:nn { enumext / enumext* }{ base-fix }
4426                 \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4427             \end{enumext}
4428         }

```

```

4429     }
4430   }
4431   {
4432     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
4433   }
4434 }

```

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

### 11.43 The command `\setenumext`

First we define a “meta families” of `\keys` to access from `\setenumext`.

```

4435 \keys_define:nn { enumext / meta-families }
4436 {
4437   enumext-1 .code:n = { \keys_set:nn { enumext / level-1 } {#1} } ,
4438   enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
4439   enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
4440   enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
4441   keyans    .code:n = { \keys_set:nn { enumext / keyans   } {#1} } ,
4442   enumext*  .code:n = { \keys_set:nn { enumext / enumext* } {#1} } ,
4443   keyans*   .code:n = { \keys_set:nn { enumext / keyans*  } {#1} } ,
4444   print*    .code:n = { \keys_set:nn { enumext / print   } { print* = {#1} } } ,
4445   print-1   .code:n = { \keys_set:nn { enumext / print   } { print-1 = {#1} } } ,
4446   print-2   .code:n = { \keys_set:nn { enumext / print   } { print-2 = {#1} } } ,
4447   print-3   .code:n = { \keys_set:nn { enumext / print   } { print-3 = {#1} } } ,
4448   print-4   .code:n = { \keys_set:nn { enumext / print   } { print-4 = {#1} } } ,
4449   print-*   .code:n = { \keys_set:nn { enumext / print   } { print-* = {#1} } } ,
4450   unknown   .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
4451 }

```

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

```

4452 \seq_const_from_clist:Nn \c__enumext_all_families_seq
4453 {
4454   enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
4455   keyans*, print-1, print-2, print-3, print-4, print-*, print*,
4456 }

```

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

```

4457 \NewDocumentCommand \setenumext { 0{enumext,1} +m }
4458 {
4459   \tl_if_novalue:nTF {#1}
4460   {
4461     \seq_map_inline:Nn \c__enumext_all_families_seq
4462   }
4463   {
4464     \seq_clear:N \l__enumext_setkey_tmpa_seq
4465     \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
4466     \int_set:Nn \l__enumext_setkey_tmpa_int
4467     {
4468       \seq_count:N \l__enumext_setkey_tmpb_seq
4469     }
4470     \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
4471     {
4472       \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
4473       \seq_map_function:NN \l__enumext_setkey_tmpb_seq \l__enumext_set_parse:n
4474       \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
4475       {
4476         \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
4477       }
4478     }
4479     {
4480       \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
4481     }
4482     \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
4483     { \seq_map_inline:Nn \c__enumext_all_families_seq }
4484     { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
4485   }
4486   {
4487     \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
4488   }
4489 }

```

(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.

```

4490 \cs_new_protected:Npn \__enumext_set_parse:n #1
4491 {
4492   \tl_set:Nc \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
4493   \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
4494   { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
4495   \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
4496   {
4497     \seq_put_right:Nc \l__enumext_setkey_tmpa_seq
4498     { \tl_trim_spaces:n {#1} }
4499   }
4500   { \__enumext_set_error:nn {#1} { } }
4501 }
4502 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
4503 { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }

```

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

## 11.44 Messages

Message used by package-load for `multicol` and `hyperref` packages.

```

4504 \msg_new:nnn { enumext } { package-load }
4505 {
4506   The ~ '#1' ~ package ~ is ~ already ~ loaded.
4507 }
4508 \msg_new:nnn { enumext } { package-not-load }
4509 {
4510   The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
4511 }
4512 \msg_new:nnn { enumext } { package-load-foot }
4513 {
4514   The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
4515 }

```

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

```

4516 \msg_new:nnn { enumext } { counters }
4517 {
4518   The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
4519   package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
4520 }

```

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

```

4521 \msg_new:nnn { enumext } { unknown-choice }
4522 {
4523   The ~ value ~ '#3' ~ for ~ '#1' ~ key ~ is ~ invalid ~ use ~ ('#2').
4524 }

```

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

```

4525 \msg_new:nnnn { enumext } { anskey-env-error }
4526 {
4527   The ~ '#1' ~ environment ~is~ reserved ~ by ~\\
4528   'enumext' ~ package, ~ It~ is~ already~ defined.
4529 }
4530 {
4531   The ~ anskey* ~ environment ~ is ~ defined ~ internally ~
4532   for ~ the ~ 'save-ans' ~ key.\\
4533 }

```

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

```

4534 \msg_new:nnn { enumext } { store-prop }
4535 {
4536   * ~ Package ~ enumext: ~ Creating ~
4537   \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
4538 }
4539 \msg_new:nnn { enumext } { store-seq }
4540 {
4541   * ~ Package ~ enumext: ~ Creating ~
4542   \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
4543 }
4544 \msg_new:nnn { enumext } { store-int }
4545 {

```



```

4546     * ~ Package ~ enumext: ~ Creating ~
4547     \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_context:.
4548 }
4549 \msg_new:nnn { enumext } { prop-seq-int-hook }
4550 {
4551     * ~ Package ~ enumext: ~ Elements ~ in ~
4552     \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
4553     * ~ Package ~ enumext: ~ Elements ~ in ~
4554     \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
4555     * ~ Package ~ enumext: ~ Value ~ off ~
4556     \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
4557 }
4558 \msg_new:nnn { enumext } { item-answer-hook }
4559 {
4560     * ~ Package ~ enumext: ~ Value ~ off ~
4561     \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\\
4562     * ~ Package ~ enumext: ~ Value ~ off ~
4563     \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\\
4564     * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
4565 }

```

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

```

4566 \msg_new:nnn { enumext } { invalid-key }
4567 {
4568     The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
4569 }
4570 \msg_new:nnn { enumext } { unknown-key-family }
4571 {
4572     Unknown~key~family~`\l_keys_key_str'~for~enumext.
4573 }

```

Messages used in length calculation.

```

4574 \msg_new:nnn { enumext } { width-negative }
4575 {
4576     Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\\
4577     The ~ key ~ '#1'~ accepts ~ values ~ >= ~ 0pt.
4578 }
4579 \msg_new:nnn { enumext } { width-zero }
4580 {
4581     Invalid ~ '#1=#2' ~ \msg_line_context:.\\
4582     The ~ key ~ '#1'~ accepts ~ values ~ > ~ 0pt.
4583 }

```

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

```

4584 \msg_new:nnn { enumext } { list-lengths }
4585 {
4586     **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
4587     \__enumext_show_length:nnn { dim } { labelsep } {#1}
4588     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4589     \__enumext_show_length:nnn { dim } { itemindent } {#1}
4590     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4591     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4592     \__enumext_show_length:nnn { dim } { listparindent } {#1}
4593     \__enumext_show_length:nnn { skip } { topsep } {#1}
4594     \__enumext_show_length:nnn { skip } { parsep } {#1}
4595     \__enumext_show_length:nnn { skip } { partopsep } {#1}
4596     \__enumext_show_length:nnn { skip } { itemsep } {#1}
4597     ****~
4598 }

```

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

```

4599 \msg_new:nnn { enumext } { list-lengths-not-nested }
4600 {
4601     **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
4602     \__enumext_show_length:nnn { dim } { labelsep } {#1}
4603     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
4604     \__enumext_show_length:nnn { dim } { itemindent } {#1}
4605     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
4606     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
4607     \__enumext_show_length:nnn { dim } { listparindent } {#1}
4608     \__enumext_show_length:nnn { skip } { topsep } {#1}
4609     \__enumext_show_length:nnn { skip } { parsep } {#1}
4610     \__enumext_show_length:nnn { skip } { partopsep } {#1}

```

```

4611     \__enumext_show_length:nnn { skip } { itemsep } {#1}
4612     *****
4613 }

```

Messages used by `ref` key.

```

4614 \msg_new:nnn { enumext } { key-ref-empty }
4615 {
4616     Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4617 }

```

Messages used by `save-ans` key.

```

4618 \msg_new:nnn { enumext } { save-ans-empty }
4619 {
4620     Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4621 }
4622 \msg_new:nnn { enumext } { save-ans-log }
4623 {
4624     * ~ Package ~ enumext: ~ Start ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
4625 }
4626 \msg_new:nnn { enumext } { save-ans-log-hook }
4627 {
4628     * ~ Package ~ enumext: ~ Stop ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
4629 }
4630 \msg_new:nnn { enumext } { save-ans-hook }
4631 {
4632     Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
4633 }

```

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

```

4634 \msg_new:nnn { enumext } { need-save-ans }
4635 {
4636     Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
4637 }
4638 \msg_new:nnn { enumext } { items-same-answer }
4639 {
4640     *****\
4641     * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~
4642     for ~ \c_left_brace_str #2 \c_right_brace_str\
4643     * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~
4644     'OK', ~ all ~ items ~ with ~ answer.\
4645     *****
4646 }
4647 \msg_new:nnn { enumext } { item-greater-answer }
4648 {
4649     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\
4650     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\
4651     Items ~ > ~ Answers.
4652 }
4653 \msg_new:nnn { enumext } { item-less-answer }
4654 {
4655     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\
4656     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\
4657     Items ~ < ~ Answers.
4658 }

```

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

```

4659 \msg_new:nnn { enumext } { missing-starred }
4660 {
4661     Missing ~ '\c_backslash_str #1*' ~ #2.
4662 }
4663 \msg_new:nnn { enumext } { many-starred }
4664 {
4665     Many ~ '\c_backslash_str #1*' ~ #2.
4666 }

```

Messages used by `\printkeyans*` command.

```

4667 \msg_new:nnn { enumext } { print-starred }
4668 {
4669     \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
4670     #2 ~ environment ~ \msg_line_context:.
4671 }

```

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

```
4672 \msg_new:nnn { enumext } { list-too-deep }
4673 {
4674   Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \\
4675   The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
4676 }
```

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

```
4677 \msg_new:nnn { enumext } { anskey-unnumber-item }
4678 {
4679   Can't ~ store ~ with ~ a ~ unnumbered ~ \c_backslash_str item ~ \msg_line_context:.
4680 }
4681 \msg_new:nnn { enumext } { anskey-already-stored }
4682 {
4683   Content ~ already ~ stored ~ for ~ this ~ \c_backslash_str item ~ \msg_line_context:.
4684 }
4685 \msg_new:nnn { enumext } { anskey-empty-arg }
4686 {
4687   Can't ~ store ~ empty ~ content ~ ~ \msg_line_context:.
4688 }
4689 \msg_new:nnn { enumext } { anskey-wrong-place }
4690 {
4691   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4692   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4693 }
4694 \msg_new:nnn { enumext } { anskey-nested }
4695 {
4696   The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
4697 }
4698 \msg_new:nnn { enumext } { anskey-math-mode }
4699 {
4700   #1 ~ can't ~ work ~ in ~ math ~ mode ~ \msg_line_context:.
4701 }
4702 \msg_new:nnn { enumext } { anskey-env-wrong }
4703 {
4704   The ~ environment ~ anskey* ~ cannot ~ use ~ in ~ '#1' ~ \msg_line_context:.
4705 }
4706 \msg_new:nnn { enumext } { ansPIC-wrong-place }
4707 {
4708   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4709   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4710 }
4711 \msg_new:nnn { enumext } { command-wrong-place }
4712 {
4713   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
4714   '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
4715 }
4716 \msg_new:nnnn { enumext } { anskey-env-key-unknown }
4717 {
4718   The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
4719   'anskey*' ~ and ~ is ~ being ~ ignored.
4720 }
4721 {
4722   The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~ '#1'.\\
4723   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
4724 }
4725 \msg_new:nnnn { enumext } { anskey-env-key-value-unknown }
4726 {
4727   The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
4728   'anskey*' ~ and ~ is ~ being ~ ignored.
4729 }
4730 {
4731   The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~ '#1'.\\
4732   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
4733 }
4734 \msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
4735 { The~key~'#1'~is~unknown~by~'\c_backslash_str anskey'~and~is~being~ignored.}
4736 {
4737   The~command~'\c_backslash_str anskey'~does~not~have~a~key~called~'#1'.\\
4738   Check~that~you~have~spelled~the~key~name~correctly.
4739 }
```

```

4740 \msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
4741 { The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.
4742 {
4743   The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~ '#1'.\\
4744   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
4745 }

```

Messages used by **keyans** and **keyanspic** environment.

```

4746 \msg_new:nnn { enumext } { keyans-nested }
4747 {
4748   The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:..
4749 }
4750 \msg_new:nnn { enumext } { keyans-wrong-level }
4751 {
4752   Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:.. \\
4753   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4754 }
4755 \msg_new:nnn { enumext } { wrong-place }
4756 {
4757   Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:.. \\
4758   '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext'.
4759 }
4760 \msg_new:nnn { enumext } { keyanspic-nested }
4761 {
4762   The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested ~ \msg_line_context:..
4763 }
4764 \msg_new:nnn { enumext } { keyanspic-wrong-level }
4765 {
4766   Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:.. \\
4767   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4768 }

```

Messages used by **\getkeyans** command.

```

4769 \msg_new:nnn { enumext } { undefined-storage-anskey }
4770 {
4771   Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:..
4772 }

```

Messages used by **\miniright** command.

```

4773 \msg_new:nnn { enumext } { missing-miniright }
4774 {
4775   Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.. \\
4776   The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
4777 }
4778 \msg_new:nnn { enumext } { wrong-miniright-place }
4779 {
4780   Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.. \\
4781   Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
4782 }
4783 \msg_new:nnn { enumext } { wrong-miniright-use }
4784 {
4785   Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.. \\
4786   '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
4787 }

```

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

```

4788 \msg_new:nnn { enumext } { nested }
4789 {
4790   The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:..
4791 }
4792 \msg_new:nnn { enumext } { item-joined }
4793 {
4794   Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~ \msg_line_context:..
4795 }
4796 \msg_new:nnn { enumext } { item-joined-columns }
4797 {
4798   Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~ \msg_line_context:..
4799 }

```

## 11.45 Finish package

Finish package implementation.

```

4800 \file_input_stop:
4801 </package>

```

## 12 Index of Implementation

The *italic* numbers denote the pages where the corresponding entry is described, the numbers underlined and all others indicate the line on which they are implemented in the package code.

Symbols	
<code>\*</code> .....	210
<code>\+</code> .....	202
<code>\-</code> .....	202
<code>\\</code> 218, 2629, 3512, 4518, 4527, 4532, 4552, 4554, 4561, 4563, 4576, 4581, 4586, 4601, 4640, 4642, 4644, 4649, 4650, 4655, 4656, 4674, 4691, 4708, 4713, 4722, 4731, 4737, 4743, 4752, 4757, 4766, 4775, 4780, 4785	
A	
above .....	<u>1423</u>
above* .....	<u>1423</u>
<code>\addvspace</code> 1077, 1105, 1221, 1300, 1363, 1369, 1397, 1414, 3351, 3366, 3468, 3483, 3777, 3791, 3832, 3846	
after .....	<u>916</u>
align .....	<u>491</u>
<code>\Alph</code> .....	36, 40, 41
<code>\Alpha</code> .....	443, 558, 603, 671, 4385
<code>\alph</code> .....	36, 40, 41
<code>\alpha</code> .....	444, 556, 4377
<code>\anskey</code> .....	12, 72, <u>2275</u>
anskey* .....	13, <u>2557</u>
<code>\anspic</code> .....	15, 98, 99, <u>3490</u>
<code>\anspic*</code> .....	67
<code>\arabic</code> .....	30, 36
<code>\arabic</code> .....	442, 555, 602, 4369, 4373, 4389
B	
base-fix .....	<u>795</u>
<code>\baselineskip</code> .....	49
<code>\baselineskip</code> .....	812, 823
before .....	<u>916</u>
before* .....	<u>916</u>
below .....	<u>1423</u>
below* .....	<u>1423</u>
bool commands:	
<code>\bool_gset_false:N</code> 318, 319, 320, 2725, 2727, 3793, 3797, 3848	
<code>\bool_gset_true:N</code> 230, 239, 1019, 1921, 1927, 3769, 3794, 3824, 3849	
<code>\bool_if:NTF</code> . 383, 395, 412, 1445, 1459, 1472, 1483, 1494, 1505, 1516, 1527, 1581, 1598, 1603, 1611, 1638, 1676, 1681, 1688, 1692, 1714, 1719, 1727, 1734, 1765, 1773, 1866, 2012, 2081, 2091, 2170, 2194, 2201, 2225, 2279, 2293, 2321, 2333, 2355, 2472, 2483, 2487, 2548, 2559, 2674, 2754, 2769, 2844, 2855, 2859, 2972, 3003, 3078, 3094, 3156, 3166, 3196, 3201, 3285, 3335, 3349, 3357, 3396, 3453, 3466, 3474, 3492, 3765, 3774, 3778, 3820, 3829, 3833, 3917, 3927, 4010, 4015, 4024, 4028, 4043, 4072, 4225, 4229, 4253, 4262, 4266, 4272, 4286, 4309	
<code>\bool_if:nTF</code> 1398, 1415, 3015, 3050, 3114, 3513, 4409	
<code>\bool_if_p:N</code> 248, 263, 808, 809, 819, 820, 1745, 1746, 1754, 1755, 1879, 1905, 1918, 1919, 1924, 1925, 2340, 2381, 2382, 2406, 2415, 2416, 2428, 2444, 2660, 2831, 2832, 2869, 2870, 3258, 3260, 3271, 3520, 3521	
<code>\bool_lazy_all:nTF</code> 246, 261, 1877, 1903, 2404, 2413, 2426, 2442, 3256, 3269	
<code>\bool_lazy_and:nnTF</code> 226, 235, 807, 818, 1744, 1753, 1917, 1923, 2339, 2346, 2380, 2512, 2524, 2659, 2665, 2830	
<code>\bool_lazy_or:nnTF</code> . . . . . 1806, 1813, 2868, 3519	
<code>\bool_new:N</code> 34, 35, 36, 37, 38, 39, 40, 41, 63, 72, 93, 98, 99, 104, 105, 108, 129, 130, 138, 139, 144, 146, 147, 161, 172, 174	
<code>\bool_not_p:n</code> 227, 236, 2341, 2347, 2431, 2446, 2661, 2666, 3259, 3272	
<code>\bool_set_eq:NN</code> . . . . . 2981, 3030, 3960, 4179	
<code>\bool_set_false:N</code> 392, 829, 1851, 1852, 1884, 1889, 1893, 1897, 1910, 2612, 3372, 3404, 3486, 3551, 3569, 3912, 3957, 4129, 4176	
<code>\bool_set_true:N</code> . 253, 254, 268, 269, 374, 378, 484, 844, 1429, 1434, 1701, 1823, 1824, 2113, 2121, 2613, 2975, 2977, 3006, 3008, 3026, 3038, 3233, 3265, 3278, 3304, 3401, 3428, 3754, 3809, 3885, 3966, 3973, 3974, 4018, 4185, 4192, 4193	
box commands:	
<code>\box_dp:N</code> . . 1117, 1121, 1125, 1136, 1140, 1151, 1160, 1166, 1176, 1189, 1195, 1201, 1232, 1233, 1234, 1237, 1247, 1251, 1260, 1267, 1272, 1280, 1309, 1310, 1313, 1320, 1333, 1341, 1347, 1355, 3581	
<code>\box_new:N</code> . . . . . 69, 167, 173	
<code>\box_set_wd:Nn</code> . . . . . 4064, 4301	
<code>\box_use_drop:N</code> . . . . . 3789, 3844, 4071, 4308	
<code>\box_wd:N</code> . . . . . 450	
C	
<code>\c</code> .....	210, 211, 708, 710, 722, 724
<code>\catcode</code> .....	2629
<code>\cB</code> .....	211
<code>\cE</code> .....	211
<code>\centering</code> .....	1400, 1417, 3602, 3782, 3837
check-ans .....	<u>1843</u>
Document class:	
article .....	42
clist commands:	
<code>\clist_const:Nn</code> .....	179
<code>\clist_map_function:nN</code> .....	3589
<code>\clist_map_inline:Nn</code> 490, 750, 849, 915, 930, 1011, 1439	
<code>\clist_map_inline:nn</code> . 48, 59, 77, 83, 95, 107, 132, 155, 178, 518, 538, 804, 854, 1025, 1545, 1790, 1857, 2060, 2078, 2110, 2401, 2763, 2930, 3143, 3146, 3173, 3183, 3186, 3206, 4493	
<code>\columnbreak</code> .....	74
<code>\columnbreak</code> .....	2343
columns .....	<u>995</u>
columns-sep .....	<u>995</u>
<code>\columnsep</code> .....	94, 97
<code>\columnsep</code> .....	3329, 3450
<code>\columnseprule</code> .....	94, 97
<code>\columnseprule</code> .....	3333, 3452
Commands provide by enumext:	
<code>\anskey</code> 28, 63, 64, 69, 71–73, 75–77, 82, 84, 93, 107, 116, 117, 122	
<code>\anspic*</code> . . . . . 28, 67, 71, 82, 83, 99–101, 116, 117	
<code>\anspic</code> .....	71, 98–101, 122
<code>\getkeyans</code> .....	71, 116, 123

`\item*` . . . . . 28, 67, 71, 82, 83, 86, 87, 108, 113, 116, 117  
`\itemwidth` . . . . . 102  
`\item` . . . . . 86, 87, 102, 107–109, 112, 113  
`\miniright` . . . . . 27, 47, 54, 55, 94, 95, 97, 98, 123  
`\printkeyans*` . . . . . 116  
`\printkeyans` . . . . . 28, 71, 116, 117  
`\setenumext` . . . . . 28, 117–120

Counters defined by `enumext`:

`enumXiii` . . . . . 26, 35  
`enumXii` . . . . . 26, 35  
`enumXiv` . . . . . 26, 35  
`enumXi` . . . . . 26, 35  
`enumXviii` . . . . . 26, 35  
`enumXvii` . . . . . 26, 35, 109  
`enumXvi` . . . . . 26, 35  
`enumXv` . . . . . 26, 35

cs commands:

`\cs_generate_variant:Nn` 452, 468, 714, 730, 2162, 2167, 2243, 2565, 3133, 3591, 4363  
`\cs_if_exist:NTF` . . . . . 422  
`\cs_if_free:NTF` . . . . . 2516, 2528  
`\cs_new:Nn` . . . . . 196  
`\cs_new:Npn` . 214, 1546, 1555, 1565, 2125, 2134, 2142  
`\cs_new_eq:NN` 345, 346, 347, 351, 352, 397, 398, 401, 402  
`\cs_new_protected:Nn` . 206, 220, 244, 277, 304, 310, 316, 322, 328, 336, 354, 369, 579, 642, 694, 805, 931, 935, 939, 943, 947, 951, 955, 959, 963, 967, 971, 975, 979, 983, 987, 991, 1026, 1038, 1062, 1079, 1090, 1107, 1182, 1206, 1223, 1285, 1302, 1324, 1359, 1365, 1440, 1454, 1468, 1479, 1490, 1501, 1512, 1523, 1609, 1712, 1725, 1742, 1763, 1791, 1796, 1821, 1862, 1872, 1915, 1930, 1937, 1946, 1951, 1956, 1961, 1970, 1975, 1980, 2002, 2168, 2192, 2199, 2223, 2230, 2291, 2392, 2503, 2566, 2602, 2633, 2657, 2699, 2723, 2752, 2767, 2795, 2828, 2864, 2876, 2884, 2935, 2939, 2958, 3011, 3046, 3062, 3072, 3088, 3226, 3254, 3283, 3290, 3313, 3343, 3355, 3394, 3418, 3436, 3461, 3472, 3509, 3553, 3567, 3587, 3592, 3608, 3627, 3744, 3763, 3799, 3818, 3876, 3899, 3906, 3915, 3925, 3942, 4083, 4120, 4142, 4148, 4161, 4217, 4321  
`\cs_new_protected:Npn` 184, 188, 192, 405, 420, 437, 447, 453, 559, 604, 676, 701, 715, 1387, 1406, 1577, 1596, 1666, 1699, 1801, 1985, 2079, 2089, 2111, 2119, 2154, 2163, 2261, 2265, 2310, 2330, 2469, 2481, 2557, 2588, 2592, 2623, 2731, 2805, 2849, 2968, 2987, 3022, 3034, 3102, 3136, 3176, 3236, 3414, 3562, 3646, 3695, 3888, 3948, 3955, 3971, 3979, 3984, 3996, 4135, 4167, 4174, 4190, 4198, 4212, 4342, 4355, 4402, 4490, 4502  
`\cs_new_protected_nopar:Nn` . . . 3935, 4059, 4154, 4296  
`\cs_new_protected_nopar:Npn` . . . . . 4002, 4245  
`\cs_set:Nn` . . . . . 2474  
`\cs_set:Npn` . . . . . 2402, 2440, 4348  
`\cs_set_eq:NN` . . 3866, 3867, 4004, 4110, 4111, 4247  
`\cs_set_protected:Nn` . . . . . 855, 871, 883, 895  
`\cs_set_protected:Npn` . 44, 53, 70, 78, 90, 96, 125, 151, 159, 469, 491, 523, 539, 586, 731, 751, 795, 831, 850, 907, 916, 995, 1012, 1423, 1534, 1782, 1843, 2022, 2061, 2097, 2394, 2756, 2919, 3134, 3174  
`\cs_to_str:N` . . . . . 439, 462  
`\cs_undefine:N` . . . . . 2505, 2506, 2507, 2508

## D

`\d` . . . . . 202  
`\DeclareDocumentEnvironment` . . . . . 358

## dim commands:

`\dim_abs:n` . . . . . 3107, 3112  
`\dim_add:Nn` . . . . . 3572  
`\dim_compare:nNnTF` . 857, 873, 885, 897, 1389, 1408, 3104, 3109, 3115, 3121, 3123, 3125, 3295, 3318, 3422, 3440, 3564, 3610, 3629, 3746, 3801  
`\dim_compare:nTF` . . . . . 2365, 2687  
`\dim_gset_eq:NN` . . . . . 3755, 3810  
`\dim_gzero:N` . . . . . 2729, 3796, 3851  
`\dim_new:N` . . . . . 66, 73, 74, 75, 92, 134, 168, 169, 175  
`\dim_set:Nn` . . 450, 845, 3001, 3107, 3112, 3114, 3117, 3118, 3122, 3124, 3127, 3128, 3130, 3298, 3321, 3424, 3442, 3594, 3612, 3619, 3631, 3638, 3681, 3730, 3748, 3803, 3998  
`\dim_set_eq:NN` 546, 593, 664, 668, 2996, 3145, 3185, 3329, 3450, 3688, 3691, 3692, 3737, 3740, 3741, 3989  
`\dim_use:N` 858, 866, 1390, 1396, 2233, 2236, 2241, 3067, 3069, 3296, 3301, 3302, 3309, 3319, 3323, 3324, 3326  
`\dim_zero:N` . . . . . 3333, 3452, 3573, 3574, 3575  
`\dim_zero_new:N` . . . . . 3625, 3644  
`\c_zero_dim` 860, 874, 886, 898, 1390, 1408, 2367, 2689, 3104, 3109, 3115, 3122, 3296, 3319, 3422, 3440, 3610, 3629, 3746, 3801

## E

`\end` . . 1393, 1411, 2196, 2227, 3348, 3365, 3465, 3482, 3767, 3790, 3822, 3845, 4411, 4420, 4427  
`\endgroup` . . . . . 2629  
`\endlist` . . . . . 346  
`\endlrbox` . . . . . 4062, 4299  
`\endminipage` . . . . . 352  
`enumext` . . . . . 5, 3207  
enumext internal commands:
 

- `\l__enumext_u_check_start_line_env_tl` . . . 32
- `\l__enumext_u_ref_the_count_tl` . . . . . 38
- `\l__enumext__resume_name_tl` . . . . . 59
- `\__enumext_add_pre_parsep:` . 48, 1036, 1038, 1038
- `\__enumext_after_args_exec:` . 45, 931, 943, 3219
- `\__enumext_after_args_exec_v:` 46, 947, 959, 3387
- `\__enumext_after_args_exec_vii:` . . . 963, 987
- `\__enumext_after_args_exec_viii:` . . . . . 991
- `\__enumext_after_env:nn` . 68, 79, 80, 96, 104, 111, 188, 188, 2643, 3375, 3772, 3827, 4097
- `\__enumext_after_hyperref:` . . . 34, 367, 369, 369
- `\__enumext_after_list:` . 95, 107, 112, 3224, 3355, 3355
- `\l__enumext_after_list_args_v_tl` . . . . . 961
- `\l__enumext_after_list_args_vii_tl` 989, 4053
- `\l__enumext_after_list_args_viii_tl` 993, 4282
- `\__enumext_after_list_v:` . . 98, 3392, 3472, 3472
- `\__enumext_after_list_vii:` . . 3874, 3906, 3906
- `\__enumext_after_list_viii:` . . 4118, 4148, 4148
- `\__enumext_after_stop_list:` . . 45, 46, 931, 939, 3370
- `\__enumext_after_stop_list_v:` 46, 947, 955, 3487
- `\l__enumext_after_stop_list_v_tl` . . . . . 957
- `\__enumext_after_stop_list_vii:` 963, 979, 3909
- `\l__enumext_after_stop_list_vii_tl` . . . 981
- `\__enumext_after_stop_list_viii:` . 983, 4151
- `\l__enumext_after_stop_list_viii_tl` . . . 985
- `\l__enumext_align_label_vii_str` . . 4045, 4049
- `\l__enumext_align_label_viii_str` . 4274, 4278
- `\l__enumext_align_label_X_str` . . . . . 159
- `\c__enumext_all_envs_clist` . . 179, 490, 750, 849, 915, 930, 1011, 1439



```

\c__enumext_all_families_seq .. 118, 4452, 4461,
    4483
\__enumext_anskey__env_keys: ..... 80
\l__enumext_anskey_env_bool 31, 78, 34, 254, 269,
    2559
\__enumext_anskey_env_clean: .. 81, 2653, 2657,
    2723
\__enumext_anskey_env_define_keys: 79, 2557,
    2566, 2637
\__enumext_anskey_env_exec: 80, 2562, 2633, 2633
\__enumext_anskey_env_keys: .. 2651, 2657, 2657
\__enumext_anskey_env_keys:n .. 79, 2585, 2588
\__enumext_anskey_env_keys:nn .... 2590, 2592
\__enumext_anskey_env_make:n 63, 78, 1826, 2557,
    2557, 2565
\__enumext_anskey_env_store: .. 81, 2652, 2657,
    2699
\__enumext_anskey_env_undefine_keys: . 79, 80,
    2602, 2654
\__enumext_anskey_env_undefine_keys:\__-
    enumext_rescan_anskey_env:n ..... 2557
\__enumext_anskey_keys:n ..... 73, 2258, 2261
\__enumext_anskey_keys:nn ..... 2263, 2265
\l__enumext_anskey_level_int .. 28, 2316, 2317
\__enumext_anskey_safe_inner:n .. 73, 74, 2281,
    2291, 2310
\__enumext_anskey_safe_outer: . 73, 2277, 2291,
    2291
\__enumext_anskey_show_wrap_arg:n . 77, 2469,
    2469, 2485, 2500
\__enumext_anskey_show_wrap_left:n 77, 2337,
    2481, 2481
\__enumext_anskey_wrapper:n ..... 2026, 2479
\__enumext_at_begin_document:n .. 33, 184, 184,
    343, 349
\l__enumext_base_line_fix_bool . 799, 809, 820,
    829
\__enumext_before_args_exec: 45, 931, 931, 3293
\__enumext_before_args_exec_v: 45, 46, 947, 947,
    3421
\__enumext_before_args_exec_vii: .. 963, 963,
    3903
\__enumext_before_args_exec_viii: 967, 4145
\__enumext_before_env:nn 79, 188, 192, 2510, 2522,
    2534, 2635
\__enumext_before_keys_exec: 45, 931, 935, 3217
\__enumext_before_keys_exec_v: .. 45, 947, 951,
    3385
\__enumext_before_keys_exec_vii ..... 963
\__enumext_before_keys_exec_vii: 46, 971, 3863
\__enumext_before_keys_exec_viii: .. 46, 975,
    4107
\__enumext_before_list: ... 94, 3211, 3290, 3290
\__enumext_before_list_v: . 97, 3380, 3418, 3418
\__enumext_before_list_vii: ... 106, 3858, 3899,
    3899
\__enumext_before_list_viii: .. 111, 4103, 4142,
    4142
\l__enumext_before_no_starred_key_v_tl 953
\l__enumext_before_no_starred_key_vii_-
    tl ..... 973
\l__enumext_before_no_starred_key_viii_-
    tl ..... 977
\l__enumext_before_starred_key_v_tl ... 949
\l__enumext_before_starred_key_vii_tl . 965
\l__enumext_before_starred_key_viii_tl 969
\__enumext_calc_hspace:NNNNNNN 90, 3102, 3102,
    3133, 3138, 3178
\__enumext_check_ans_active: 65, 94, 1862, 1862,
    3294, 3902
\g__enumext_check_ans_item_tl ..... 84
\g__enumext_check_ans_key_bool .. 66, 138, 318,
    1921, 1927, 2012
\l__enumext_check_ans_key_bool 66, 86, 87, 1847,
    1852, 1918, 1924
\__enumext_check_ans_key_hook: 66, 1915, 1915,
    3369, 3910
\__enumext_check_ans_level: 65, 1862, 1868, 1872
\__enumext_check_ans_log: 66, 67, 1961, 1961, 2016
\__enumext_check_ans_log_msg_greater: 1961,
    1967, 1980
\__enumext_check_ans_log_msg_less: 1961, 1965,
    1970
\__enumext_check_ans_log_msg_same_ok: 1961,
    1966, 1975
\__enumext_check_ans_msg_greater: 1937, 1943,
    1956
\__enumext_check_ans_msg_less: 1937, 1941, 1946
\__enumext_check_ans_msg_same_ok: 1937, 1942,
    1951
\__enumext_check_ans_show: .. 66, 67, 1937, 1937,
    2014
\g__enumext_check_ans_show_bool ..... 95
\l__enumext_check_answers_bool 63, 65, 73, 138,
    1824, 1851, 1866, 2170, 2194, 2201, 2225, 2279, 2844,
    2972, 3003, 4015
\__enumext_check_starred_cmd:n 32, 67, 84, 1985,
    1985, 3390, 3548, 4116
\g__enumext_check_starred_cmd_int 138, 1988,
    1994, 1999, 3044, 3518, 4224
\l__enumext_check_start_line_env_tl 138, 283,
    290, 297, 1991, 1997, 2000
\l__enumext_columns_sep_v_dim 3440, 3442, 3450
\l__enumext_columns_sep_vii_dim .. 3610, 3612,
    3621, 3685, 4081
\l__enumext_columns_sep_viii_dim . 3629, 3631,
    3640, 3734, 4319
\l__enumext_columns_v_int 1228, 3438, 3446, 3458,
    3463
\l__enumext_columns_vii_int .. 3615, 3618, 3622,
    3649, 3653, 3656, 3662, 3668, 3672, 4076, 4087
\l__enumext_columns_viii_int . 3634, 3637, 3641,
    3698, 3702, 3705, 3711, 3717, 3721, 4314, 4327
\l__enumext_counter_i_tl ..... 44, 429
\l__enumext_counter_ii_tl ..... 44, 430
\l__enumext_counter_iii_tl ..... 44, 431
\l__enumext_counter_iv_tl ..... 44, 432
\c__enumext_counter_style_tl ..... 30, 49, 208
\g__enumext_counter_styles_tl . 27, 36, 66, 440,
    458
\l__enumext_counter_v_tl ..... 44, 433, 684
\l__enumext_counter_vi_tl ..... 44, 434
\l__enumext_counter_vii_tl ..... 44, 435, 614
\l__enumext_counter_viii_tl ..... 44, 436, 631
\l__enumext_current_widest_dim 27, 66, 464, 547,
    594, 665, 669
\__enumext_default_item:n ... 2968, 2968, 3019
\__enumext_define_counters:Nn 26, 420, 420, 429,
    430, 431, 432, 433, 434, 435, 436

```



\\_\_enumext\_endminipage: . 33, [349](#), 352, 364, 3604, 4061, [4298](#)  
 \g\_\_enumext\_envir\_name\_tl 31, [34](#), 255, 270, 326, 1794, 1799, 1809, 1949, 1954, 1959, 1973, 1978, 1983  
 \\_\_enumext\_execute\_after\_env: 32, 33, 63, 66, 67, 77, [2002](#), 2002, 3375, 4097  
 \\_\_enumext\_fake\_item: . . . . . [855](#), 855, 3165  
 \l\_\_enumext\_fake\_item\_indent\_v\_dim 874, 879  
 \l\_\_enumext\_fake\_item\_indent\_v\_tl 876, 3027, 3031, 3039  
 \l\_\_enumext\_fake\_item\_indent\_vii\_dim 886, 891  
 \l\_\_enumext\_fake\_item\_indent\_vii\_tl 888, 4057  
 \l\_\_enumext\_fake\_item\_indent\_viii\_dim . 898, 903, 4290  
 \l\_\_enumext\_fake\_item\_indent\_viii\_tl . . 900, 4288, 4293  
 \l\_\_enumext\_fake\_item\_indent\_X\_tl . . . . . [96](#)  
 \\_\_enumext\_fake\_item\_vii: . . . . . [855](#), 883, 3195  
 \\_\_enumext\_fake\_item\_viii: . . . . . [855](#), 895, 3200  
 \\_\_enumext\_filter\_save\_key:n . . 70, 2086, 2094, 2117, 2123, [2125](#), 2125, 4367, 4371, 4375, 4379, 4383, 4387  
 \\_\_enumext\_filter\_save\_key\_key:n . . 70, [2125](#), 2130, 2134  
 \\_\_enumext\_filter\_save\_key\_pair:nn 70, [2125](#), 2131, 2142  
 \\_\_enumext\_filter\_series:n 58, [1546](#), 1546, 1589, 1601, 1606  
 \\_\_enumext\_filter\_series\_key:n 58, [1546](#), 1551, 1555  
 \\_\_enumext\_filter\_series\_pair:nn . . 58, [1546](#), 1552, 1565  
 \g\_\_enumext\_footnote\_arg\_seq . [156](#), 2941, 2954, 2964  
 \g\_\_enumext\_footnote\_int . [156](#), 2948, 2951, 2953, 2955  
 \g\_\_enumext\_footnote\_int\_seq . [156](#), 2942, 2955, 2960, 2963  
 \\_\_enumext\_footnotes\_key\_bool . . . . . 34  
 \l\_\_enumext\_footnotes\_key\_bool 29, 34, 109, [146](#), 378, 383, 392, 4024, 4072, 4262, 4309  
 \\_\_enumext\_footnotetext:nn . . . 2935, 2935, 2965  
 \\_\_enumext\_getkeyans:nn . . 116, 4351, [4355](#), 4355  
 \\_\_enumext\_getkeyans\_aux:n 116, 4339, [4342](#), 4342  
 \l\_\_enumext\_hyperref\_bool 29, 34, [146](#), 374, 395, 412, 2382, 2832, 4010, 4253  
 \\_\_enumext\_hypertarget:nn 34, [369](#), 397, 401, 417  
 \\_\_enumext\_if\_is\_int:n . . . . . 200  
 \\_\_enumext\_if\_is\_int:nTF . . . . . [200](#), 703, 717  
 \\_\_enumext\_internal\_mini\_page: . . 33, [354](#), 354, 3228, 3878  
 \\_\_enumext\_is\_not\_nested: . 26, 31, 92, [220](#), 220, 3229, 3879  
 \\_\_enumext\_is\_on\_first\_level: . 26, 31, 92, [220](#), 244, 3234, 3886  
 \g\_\_enumext\_item\_anskey\_int 73, 84, [138](#), 313, 340, 341, 1934, 2286, 2846  
 \\_\_enumext\_item\_answer\_diff: 66, 67, [1930](#), 1930, 2009  
 \g\_\_enumext\_item\_answer\_diff\_int 66, [138](#), 314, 1932, 1939, 1963  
 \l\_\_enumext\_item\_column\_pos\_vii\_int 107, 3656, 3662, 3668, 3672, 3679, 3938, 4076, 4079  
 \l\_\_enumext\_item\_column\_pos\_viii\_int . . 112, 3705, 3711, 3717, 3721, 3728, 4157, 4314, 4317  
 \\_\_enumext\_item\_column\_pos\_X\_int . . . . . [159](#)  
 \g\_\_enumext\_item\_count\_all\_vii\_int 107, 3680, 3939, 4087, 4094  
 \g\_\_enumext\_item\_count\_all\_viii\_int 112, 3729, 4158, 4326, 4334  
 \g\_\_enumext\_item\_count\_all\_X\_int . . . . . [159](#)  
 \g\_\_enumext\_item\_number\_bool . . . . . [138](#)  
 \l\_\_enumext\_item\_number\_bool 65, 144, 1884, 1889, 1893, 1897, 1910, 2321, 2548, 2975, 3006, 4018  
 \g\_\_enumext\_item\_number\_int . . 65, [138](#), 312, 339, 341, 1883, 1888, 1892, 1896, 1909, 1934, 2974, 3005, 4017  
 \\_\_enumext\_item\_peek\_args\_vii: 107, 108, 3940, 3942, 3942  
 \\_\_enumext\_item\_peek\_args\_viii: . . 112, 4159, 4161, 4161  
 \\_\_enumext\_item\_starred: . . 89, [3062](#), 3062, 3080  
 \l\_\_enumext\_item\_starred\_vii\_bool 3957, 3973, 4028  
 \l\_\_enumext\_item\_starred\_viii\_bool 4176, 4192, 4266, 4286  
 \l\_\_enumext\_item\_starred\_X\_bool . . . . . [159](#)  
 \\_\_enumext\_item\_std:w . . 33, 86–88, 101, [343](#), 347, 2978, 2984, 3009, 3027, 3031, 3039, 3585  
 \g\_\_enumext\_item\_symbol\_aux\_vii\_tl 3981, 4030, 4033, 4037, 4039  
 \g\_\_enumext\_item\_symbol\_aux\_X\_tl . . . . . [159](#)  
 \l\_\_enumext\_item\_symbol\_sep\_vii\_dim . . 3990, 3998, 4036, 4038  
 \g\_\_enumext\_item\_symbol\_tl . . . 87, [60](#), [122](#), 2993, 3068, 3085  
 \l\_\_enumext\_item\_symbol\_vii\_tl . . . . . 4033  
 \l\_\_enumext\_item\_text\_vii\_box 4023, 4064, 4071  
 \l\_\_enumext\_item\_text\_viii\_box 4261, 4301, 4308  
 \l\_\_enumext\_item\_text\_X\_box . . . . . [159](#)  
 \l\_\_enumext\_item\_width\_vii\_dim . . 3619, 3683, 3691, 3692  
 \l\_\_enumext\_item\_width\_viii\_dim . . 3638, 3732, 3740, 3741  
 \l\_\_enumext\_item\_width\_X\_dim . . . . . [159](#)  
 \l\_\_enumext\_itemindent\_X\_dim . . . . . [70](#)  
 \l\_\_enumext\_itemsep\_vii\_skip . . . . . 4093  
 \l\_\_enumext\_itemsep\_viii\_skip . . . . . 4333  
 \l\_\_enumext\_joined\_item\_aux\_vii\_int . . 3677, 3678, 3679, 3680, 3686  
 \l\_\_enumext\_joined\_item\_aux\_viii\_int . 3726, 3727, 3728, 3729, 3735  
 \l\_\_enumext\_joined\_item\_aux\_X\_int . . . . . [159](#)  
 \\_\_enumext\_joined\_item\_vii:w . . 108, 3945, 3946, 3948, 3948  
 \l\_\_enumext\_joined\_item\_vii\_int . . 3648, 3649, 3652, 3654, 3660, 3665, 3670, 3675, 3677, 3683  
 \\_\_enumext\_joined\_item\_viii:w . 112, 4164, 4165, 4167, 4167  
 \l\_\_enumext\_joined\_item\_viii\_int . 3697, 3698, 3701, 3703, 3709, 3714, 3719, 3724, 3726, 3732  
 \l\_\_enumext\_joined\_item\_X\_int . . . . . [159](#)  
 \l\_\_enumext\_joined\_width\_vii\_dim . 3681, 3688, 3691, 4054, 4066  
 \l\_\_enumext\_joined\_width\_viii\_dim 3730, 3737, 3740, 4283, 4303  
 \l\_\_enumext\_joined\_width\_X\_dim . . . . . [159](#)  
 \\_\_enumext\_keyans\_addto\_prop:n 82, [2731](#), 2731, 3041, 3515

```

\__enumext_keyans_addto_seq:n . 83, 2805, 2805,
    3043, 3517
\__enumext_keyans_addto_seq_link: 2805, 2826,
    2828, 4223
\__enumext_keyans_anspic_code:nnn . 99, 3506,
    3509, 3509
\__enumext_keyans_default_item:n . . 87, 3022,
    3022, 3058
\l__enumext_keyans_env_bool 34, 3259, 3272, 3401,
    3486
\__enumext_keyans_fake_item: . . 855, 871, 3155
\l__enumext_keyans_level_h_int . . 28, 624, 651,
    2301, 2540, 2783, 4122, 4123
\l__enumext_keyans_level_int . . 28, 1381, 2297,
    2536, 2778, 3400, 3405, 3500
\__enumext_keyans_make_label: 36, 89, 3088, 3088,
    3153
\__enumext_keyans_mini_addvspace: 52, 97, 1285,
    1285, 3430
\__enumext_keyans_mini_right_cmd:n 55, 1383,
    1406, 1406
\__enumext_keyans_mini_set_vskip: . 52, 1223,
    1223, 1287
\__enumext_keyans_multi_addvspace: 97, 1079,
    1090, 3455
\__enumext_keyans_multi_set_vskip: 49, 1079,
    1079, 1092
\__enumext_keyans_multicols_start: 97, 3434,
    3436, 3436
\__enumext_keyans_multicols_stop: . 98, 1410,
    3461, 3461, 3485
\__enumext_keyans_parse_keys:n 3379, 3414, 3414
\l__enumext_keyans_pic_above_int . 133, 3595,
    3596, 3598
\l__enumext_keyans_pic_above_skip . 101, 133,
    3539, 3579
\__enumext_keyans_pic_arg_two: 100, 3537, 3567,
    3567
\l__enumext_keyans_pic_below_int . 133, 3595,
    3596, 3599
\l__enumext_keyans_pic_body_seq . 99–101, 133,
    3504, 3544, 3603
\__enumext_keyans_pic_do:n 101, 3544, 3546, 3587,
    3587, 3591
\l__enumext_keyans_pic_level_int . . 28, 1373,
    2305, 2544, 2734, 2773, 2808, 2886, 3555, 3556
\__enumext_keyans_pic_row:n . . . 101, 3589, 3592,
    3592
\__enumext_keyans_pic_safe_exec: . 100, 3533,
    3553, 3553
\__enumext_keyans_pic_skip_abs:N . 100, 3562,
    3562, 3578
\l__enumext_keyans_pic_width_dim . 133, 3594,
    3601
\__enumext_keyans_redefine_item: . . 88, 3046,
    3046, 3152
\__enumext_keyans_ref: . . . . . 40, 676, 694, 3154
\__enumext_keyans_ref:n . . . . . 40, 673, 676, 676
\__enumext_keyans_safe_exec: . 3378, 3394, 3394
\__enumext_keyans_show_ans: . . 2849, 2857, 2876
\__enumext_keyans_show_item_opt: . 2849, 2864,
    3039, 3529, 4289
\__enumext_keyans_show_left:n . 88, 2849, 2849,
    3037, 3524
\__enumext_keyans_show_pos: . . 2849, 2861, 2884
\__enumext_keyans_starred_item:n . . 88, 3034,
    3034, 3054
\__enumext_keyans_start_line: . 26, 32, 277, 277,
    3402, 3560, 4127
\__enumext_keyans_store_ref: . . 82, 2752, 2752,
    3042, 3516, 4221
\__enumext_keyans_store_ref_aux_i: 82, 2752,
    2764, 2767
\__enumext_keyans_store_ref_aux_ii: 83, 2752,
    2793, 2795
\__enumext_keyans_wrapper_opt:n . . 2029, 2872
\l__enumext_label_copy_i_tl . . 2436, 2771, 2776,
    2781, 2786
\l__enumext_label_copy_v_tl . . . . . 2781
\l__enumext_label_copy_vi_tl . . . . . 2776
\l__enumext_label_copy_vii_tl 2411, 2422, 2453,
    2771
\l__enumext_label_copy_viii_tl . . . . . 2786
\l__enumext_label_copy_X_tl . . . . . 148
\l__enumext_label_fill_left_v_tl . . . . . 3092
\l__enumext_label_fill_left_X_tl . . . . . 96
\l__enumext_label_fill_right_v_tl . . . . . 3099
\l__enumext_label_fill_right_X_tl . . . . . 96
\l__enumext_label_font_style_v_tl 3093, 3528
\l__enumext_label_font_style_vii_tl . . 4042
\l__enumext_label_font_style_viii_tl . . 4271
\l__enumext_label_i_tl . . . . . 539
\l__enumext_label_ii_tl . . . . . 539
\l__enumext_label_iii_tl . . . . . 539
\l__enumext_label_iv_tl . . . . . 539
\__enumext_label_style:Nnn 26, 36, 453, 453, 468,
    544, 591, 662, 666
\l__enumext_label_v_tl . . 82, 83, 659, 2739, 2813,
    2878, 2913, 3036, 3040, 3382, 3523, 3525
\l__enumext_label_vi_tl . 82, 83, 659, 2736, 2810,
    3523, 3525, 3529
\l__enumext_label_vii_tl . 586, 3968, 3993, 4000
\l__enumext_label_viii_tl 586, 4187, 4215, 4219
\l__enumext_label_width_by_box . . 66, 449, 450
\__enumext_label_width_by_box:Nn 36, 447, 447,
    452, 464, 727
\l__enumext_labelsep_i_dim . . . 2881, 2916, 4227,
    4242
\l__enumext_labelsep_v_dim . . . . . 3445
\l__enumext_labelsep_vii_dim . 3614, 3623, 3684,
    3991, 4052, 4068
\l__enumext_labelsep_viii_dim 3633, 3642, 3733,
    4281, 4290, 4305
\l__enumext_labelwidth_i_dim . 2881, 2916, 4227,
    4242
\l__enumext_labelwidth_v_dim . . . . . 3445
\l__enumext_labelwidth_vii_dim . . 3614, 3622,
    3684, 4045, 4049, 4067
\l__enumext_labelwidth_viii_dim . . 3633, 3641,
    3733, 4274, 4278, 4304
\l__enumext_leftmargin_tmp_v_bool . 100, 3569
\l__enumext_leftmargin_tmp_X_bool . . . . . 70
\l__enumext_leftmargin_tmp_X_dim . . . . . 70
\l__enumext_leftmargin_X_dim . . . . . 70
\__enumext_level: 196, 196, 568, 571, 572, 581, 583,
    858, 862, 866, 933, 937, 941, 945, 1028, 1030, 1032,
    1034, 1067, 1069, 1071, 1073, 1077, 1110, 1113, 1132,
    1141, 1147, 1152, 1156, 1167, 1171, 1172, 1177, 1213,

```

1217, 1390, 1396, 1443, 1445, 1447, 1450, 1457, 1459,  
 1461, 1464, 2081, 2083, 2085, 2113, 2114, 2116, 2172,  
 2180, 2184, 2188, 2474, 2477, 2478, 2977, 2978, 2982,  
 2983, 2984, 2991, 2993, 2997, 2998, 3001, 3008, 3009,  
 3064, 3067, 3069, 3076, 3077, 3078, 3081, 3084, 3214,  
 3216, 3265, 3278, 3285, 3296, 3298, 3301, 3302, 3304,  
 3309, 3316, 3319, 3321, 3323, 3324, 3325, 3326, 3329,  
 3335, 3340, 3346, 3349, 3351, 3357  
 \l\_\_enumext\_level\_h\_int 28, 228, 250, 264, 607, 644,  
 1880, 1900, 2430, 2447, 2514, 2526, 3273, 3880, 3881  
 \l\_\_enumext\_level\_int . 92, 28, 198, 237, 249, 265,  
 356, 1040, 1184, 1377, 1874, 1906, 2004, 2407, 2417,  
 2423, 2429, 2437, 2445, 2452, 2513, 2525, 3168, 3230,  
 3231, 3241, 3249, 3263, 3276, 3331, 3409, 3496, 3919,  
 3929, 4130  
 \\_\_enumext\_list\_arg\_two\_i: . . . . . 3134  
 \\_\_enumext\_list\_arg\_two\_ii: . . . . . 3134  
 \\_\_enumext\_list\_arg\_two\_iii: . . . . . 3134  
 \\_\_enumext\_list\_arg\_two\_iv: . . . . . 3134  
 \\_\_enumext\_list\_arg\_two\_v: . 88, 3134, 3384, 3570  
 \\_\_enumext\_list\_arg\_two\_vii: . . . . 3174, 3862  
 \\_\_enumext\_list\_arg\_two\_viii: . . . . 3174, 4106  
 \l\_\_enumext\_listoffset\_v\_dim . . . . . 3447  
 \l\_\_enumext\_listparindent\_vii\_dim . . . 4055  
 \l\_\_enumext\_listparindent\_viii\_dim . . 4284  
 \\_\_enumext\_log\_answer\_vars: . 33, 328, 336, 2011  
 \\_\_enumext\_log\_global\_vars: . 32, 328, 328, 2010  
 \\_\_enumext\_make\_label: 36, 86, 87, 89, 3072, 3072,  
 3163  
 \l\_\_enumext\_mark\_answer\_sym\_tl 72, 2035, 2238,  
 2489, 2888, 2901, 4231  
 \l\_\_enumext\_mark\_position\_str 122, 2039, 2040,  
 2066, 2067, 2236  
 \l\_\_enumext\_mark\_ref\_sym\_tl . . 2052, 2387, 2840  
 \\_\_enumext\_mini\_addvspace: . . 51, 94, 1206, 1206,  
 3306  
 \\_\_enumext\_mini\_addvspace\_vii: 54, 1359, 1359,  
 3758  
 \\_\_enumext\_mini\_addvspace\_viii: 54, 1359, 1365,  
 3813  
 \_\_enumext\_mini\_env\* . . . . . 354  
 \\_\_enumext\_mini\_right\_cmd:n . 54, 55, 1385, 1387,  
 1387  
 \\_\_enumext\_mini\_set\_vskip: . 49, 1107, 1107, 1208  
 \\_\_enumext\_mini\_set\_vskip\_vii: 53, 1302, 1302,  
 1361  
 \\_\_enumext\_mini\_set\_vskip\_viii: 53, 1302, 1324,  
 1367  
 \\_\_enumext\_minipage:w 33, 349, 351, 360, 3601, 4054,  
 4283  
 \l\_\_enumext\_minipage\_active\_v\_bool . . 97, 98,  
 3428, 3453, 3466, 3474  
 \g\_\_enumext\_minipage\_active\_vii\_bool . . 104,  
 3769, 3774, 3793  
 \l\_\_enumext\_minipage\_active\_vii\_bool . 3754,  
 3765  
 \g\_\_enumext\_minipage\_active\_viii\_bool 3824,  
 3829, 3848  
 \l\_\_enumext\_minipage\_active\_viii\_bool 3809,  
 3820  
 \g\_\_enumext\_minipage\_active\_X\_bool . . . 159  
 \l\_\_enumext\_minipage\_active\_X\_bool . . . . 84  
 \g\_\_enumext\_minipage\_after\_skip 84, 1306, 1318,  
 3791, 3846  
 \l\_\_enumext\_minipage\_after\_skip 50, 51, 95, 98,  
 84, 1123, 1138, 1158, 1174, 1189, 1195, 1201, 1215,  
 1225, 1234, 1237, 1249, 1267, 1278, 1294, 1326, 1339,  
 1353, 3366, 3483  
 \g\_\_enumext\_minipage\_center\_vii\_bool . 3778,  
 3794  
 \g\_\_enumext\_minipage\_center\_viii\_bool 3833,  
 3849  
 \g\_\_enumext\_minipage\_center\_X\_bool . . . 159  
 \l\_\_enumext\_minipage\_hsep\_v\_dim . . . 97, 3426  
 \l\_\_enumext\_minipage\_hsep\_vii\_dim . . . . 3752  
 \l\_\_enumext\_minipage\_hsep\_viii\_dim . . . 3807  
 \l\_\_enumext\_minipage\_left\_skip 50, 97, 84, 1115,  
 1130, 1149, 1164, 1211, 1221, 1226, 1232, 1241, 1258,  
 1270, 1290, 1300, 1304, 1309, 1313, 1327, 1331, 1345,  
 1363, 1369  
 \l\_\_enumext\_minipage\_left\_v\_dim 97, 3424, 3432  
 \l\_\_enumext\_minipage\_left\_vii\_dim 3748, 3760  
 \l\_\_enumext\_minipage\_left\_viii\_dim 3803, 3815  
 \l\_\_enumext\_minipage\_left\_X\_dim . . . . . 84  
 \g\_\_enumext\_minipage\_right\_skip 84, 1305, 1310,  
 1314, 3777, 3832  
 \l\_\_enumext\_minipage\_right\_skip . 50, 84, 1119,  
 1134, 1154, 1169, 1227, 1233, 1245, 1263, 1274, 1328,  
 1335, 1349, 1397, 1414  
 \l\_\_enumext\_minipage\_right\_v\_dim . . 97, 1408,  
 1413, 3422, 3426  
 \g\_\_enumext\_minipage\_right\_vii\_dim 104, 3756,  
 3776, 3796  
 \l\_\_enumext\_minipage\_right\_vii\_dim 104, 3746,  
 3751, 3757  
 \g\_\_enumext\_minipage\_right\_viii\_dim . . 3811,  
 3831, 3851  
 \l\_\_enumext\_minipage\_right\_viii\_dim . . 3801,  
 3806, 3812  
 \g\_\_enumext\_minipage\_right\_X\_dim . . . . . 159  
 \g\_\_enumext\_minipage\_right\_X\_skip . . . . 159  
 \g\_\_enumext\_minipage\_stat\_int . 94, 97, 84, 1402,  
 1419, 3305, 3359, 3364, 3429, 3476, 3481  
 \l\_\_enumext\_miniright\_code\_vii\_box 3785, 3789  
 \g\_\_enumext\_miniright\_code\_vii\_tl 104, 3780,  
 3787, 3795  
 \l\_\_enumext\_miniright\_code\_viii\_box . . 3840,  
 3844  
 \g\_\_enumext\_miniright\_code\_viii\_tl 3835, 3842,  
 3850  
 \l\_\_enumext\_miniright\_code\_X\_box . . . . . 159  
 \\_\_enumext\_multi\_addvspace: . 48, 95, 1062, 1062,  
 3337  
 \\_\_enumext\_multi\_set\_vskip: 47, 1026, 1026, 1064  
 \l\_\_enumext\_multicols\_above\_ii\_skip . . 1045  
 \l\_\_enumext\_multicols\_above\_iii\_skip . . 1051  
 \l\_\_enumext\_multicols\_above\_iv\_skip . . 1057  
 \l\_\_enumext\_multicols\_above\_v\_skip 1081, 1095,  
 1105  
 \l\_\_enumext\_multicols\_above\_X\_skip . . . . 78  
 \l\_\_enumext\_multicols\_below\_v\_skip 1085, 1099,  
 3468  
 \l\_\_enumext\_multicols\_below\_X\_skip . . . . 78  
 \\_\_enumext\_multicols\_start: 94, 3311, 3313, 3313  
 \\_\_enumext\_multicols\_stop: 95, 1392, 3343, 3343,  
 3368  
 \\_\_enumext\_nested\_base\_line\_fix: 42, 795, 805,  
 3245, 3896

```

\__enumext_newlabel:nn    29, 34, 76, 405, 405, 2463,
    2799
\l__enumext_newlabel_arg_one_tl    29, 34, 76, 82,
    148, 2386, 2456, 2464, 2788, 2800, 2838
\l__enumext_newlabel_arg_two_tl    29, 35, 75, 148,
    2410, 2420, 2434, 2450, 2465, 2775, 2780, 2785, 2801
\__enumext_parse_keys:n    42, 58, 3210, 3236, 3236
\__enumext_parse_keys_vii:n    . 42, 58, 3856, 3888,
    3888
\__enumext_parse_keys_viii:n    . 4101, 4135, 4135
\__enumext_parse_save_key:n    70, 2106, 2111, 2111
\__enumext_parse_save_key_vii:n    70, 2101, 2111,
    2119
\__enumext_parse_series:n    . . . . . 106
\__enumext_parse_series:n    . . 58, 93, 1577, 1577,
    3244, 3894
\__enumext_parse_store_keys:n    . . . . . 93
\l__enumext_parsep_i_skip    1043, 1045, 1187, 1235
\l__enumext_parsep_ii_skip    . . 1049, 1051, 1193
\l__enumext_parsep_iii_skip    . . 1055, 1057, 1199
\l__enumext_parsep_vii_skip    . . . . . 4056
\l__enumext_parsep_viii_skip    . . . . . 4285
\l__enumext_partopsep_v_skip    . 1097, 1101, 1261,
    1265, 1272, 1276, 1292, 1296
\l__enumext_partopsep_viii_skip    . . . . . 1337
\__enumext_phantomsection:    34, 369, 398, 402, 418
\__enumext_print_footnote:    . . . 2935, 2958, 4074,
    4311
\__enumext_print_keyans_box:NN    72, 2230, 2230,
    2243, 2476, 2880, 2915, 4227, 4242
\l__enumext_print_keyans_i_tl    . . . . 4372, 4394
\l__enumext_print_keyans_ii_tl    . . 4376, 4395
\l__enumext_print_keyans_iii_tl    . . 4380, 4396
\l__enumext_print_keyans_iv_tl    . . 4384, 4397
\l__enumext_print_keyans_starred_tl    116, 117,
    122, 4368, 4416
\l__enumext_print_keyans_vii_tl    116, 4388, 4398
\l__enumext_print_keyans_X_tl    . . . . . 122
\__enumext_printkeyans:nnn    117, 4399, 4402, 4402
\__enumext_redefine_item:    . 87, 3011, 3011, 3162
\l__enumext_ref_key_arg_tl    38, 49, 211, 561, 562,
    575, 606, 609, 620, 626, 637, 678, 679, 690
\l__enumext_ref_the_count_tl    . 38, 49, 568, 571,
    574, 614, 616, 619, 631, 633, 636, 684, 686, 689
\__enumext_regex_counter_style:    . . 30, 38, 206,
    206, 569, 615, 632, 685
\__enumext_register_counter_style:Nn    . . 437,
    437, 442, 443, 444, 445, 446
\__enumext_remove_extra_parsep_vii:    . . 3871,
    4083, 4083
\__enumext_remove_extra_parsep_viii:    . 4115,
    4321, 4321
\__enumext_renew_footnote:    . . . 2935, 2939, 4026,
    4264
\l__enumext_renew_the_count_v_tl    687, 696, 698
\l__enumext_renew_the_count_vii_tl    617, 646,
    648
\l__enumext_renew_the_count_viii_tl    634, 653,
    655
\l__enumext_renew_the_count_X_tl    . . . . . 49
\__enumext_rescan_anskey_env:n    . . 80, 81, 2623,
    2709, 2717
\__enumext_reset_global_bool:    . . 304, 307, 316
\__enumext_reset_global_int:    . . . 304, 306, 310
\__enumext_reset_global_tl:    . . . . 304, 308, 322
\__enumext_reset_global_vars:    . 32, 67, 304, 304,
    2019
\l__enumext_resume_active_bool    58, 61, 60, 1581,
    1701
\__enumext_resume_counter:    . . 60, 61, 1699, 1705,
    1712
\__enumext_resume_counter:n    . 58, 61, 1670, 1675,
    1699, 1699, 1769, 1777
\__enumext_resume_counter_save_ans:    . . 61, 62,
    1699, 1710, 1742
\__enumext_resume_counter_series:    . 61, 1699,
    1708, 1725
\g__enumext_resume_int    . . . 60, 1622, 1716, 1717
\__enumext_resume_last:n    58, 59, 1577, 1583, 1596
\l__enumext_resume_name_tl    60, 1618, 1626, 1629,
    1645, 1653, 1656, 1702, 1703, 1731, 1738
\__enumext_resume_save_counter:    59, 1609, 1609,
    3373, 3913
\__enumext_resume_series:n    . 60, 1540, 1666, 1666
\__enumext_resume_starred:    . 62, 1541, 1763, 1763
\g__enumext_resume_vii_int    . 107, 60, 1649, 1721,
    1722
\__enumext_safe_exec:    . . 33, 92, 3209, 3226, 3226
\__enumext_safe_exec_vii:    . 33, 3855, 3876, 3876
\__enumext_safe_exec_viii:    . . . 4100, 4120, 4120
\l__enumext_series_name_tl    . . . . . 61
\l__enumext_series_str    . 59, 93, 1538, 1579, 1587,
    1588, 1590, 1592, 1613, 1616, 1620, 1640, 1643, 1647,
    3240, 3892
\__enumext_set_error:nn    . . . . . 4490, 4500, 4502
\__enumext_set_parse:n    . . . . . 4473, 4490, 4490
\l__enumext_setkey_tmpa_int    . . . 117, 4466, 4470
\l__enumext_setkey_tmpa_seq    . . 117, 4464, 4474,
    4480, 4482, 4484, 4497
\l__enumext_setkey_tmpa_tl    . . . . 117, 4472, 4476
\l__enumext_setkey_tmpb_seq    . . 117, 4465, 4468,
    4472, 4473
\l__enumext_setkey_tmpb_tl    117, 4492, 4494, 4495
\l__enumext_show_answer_bool    . 2046, 2070, 2483,
    2855, 2869, 3520, 4225
\__enumext_show_length:nnn    . . 44, 214, 214, 4587,
    4588, 4589, 4590, 4591, 4592, 4593, 4594, 4595, 4596,
    4602, 4603, 4604, 4605, 4606, 4607, 4608, 4609, 4610,
    4611
\l__enumext_show_position_bool    . . 2049, 2073,
    2487, 2859, 2870, 3521, 4229
\g__enumext_standar_bool    31, 92, 34, 227, 230, 248,
    319, 1611, 1676, 1688, 1714, 1727, 1765, 1905, 1919,
    3260
\l__enumext_standar_bool    . 92, 95, 34, 2415, 2428,
    2444, 3233, 3372
\l__enumext_standar_first_bool    31, 92, 34, 253,
    808, 1598, 1745, 1807, 1814
\__enumext_standar_item_vii:w    . 108, 3953, 3955,
    3955
\__enumext_standar_item_viii:w    112, 113, 4172,
    4174, 4174
\__enumext_standar_ref:    . . . . 38, 559, 579, 3164
\__enumext_standar_ref:n    . . . . 38, 551, 559, 559
\g__enumext_standar_series_tl    . 60, 1600, 1601,
    1767, 1770
\g__enumext_starred_bool    31, 106, 107, 34, 236, 239,
    263, 320, 1638, 1681, 1692, 1719, 1734, 1773, 1879,

```



1925, 2406, 2416, 2446, 2769, 3797  
 \l\_\_enumext\_starred\_bool 106, 107, 34, 2341, 2347,  
 2431, 2472, 2661, 2666, 3885, 3912  
 \\_\_enumext\_starred\_columns\_set\_vii: .. 3608,  
 3608, 3857  
 \\_\_enumext\_starred\_columns\_set\_viii: . 3608,  
 3627, 4102  
 \l\_\_enumext\_starred\_first\_bool 31, 34, 268, 819,  
 1603, 1754, 1807, 1814  
 \\_\_enumext\_starred\_item:nn ... 2987, 2987, 3017  
 \\_\_enumext\_starred\_item\_exec: . 113, 4217, 4217,  
 4268  
 \\_\_enumext\_starred\_item\_vii:w . 108, 3952, 3971,  
 3971  
 \\_\_enumext\_starred\_item\_vii\_aux\_i:w .. 3971,  
 3976, 3979  
 \\_\_enumext\_starred\_item\_vii\_aux\_ii:w . 3971,  
 3977, 3982, 3984  
 \\_\_enumext\_starred\_item\_vii\_aux\_iii:w 3971,  
 3987, 3996  
 \\_\_enumext\_starred\_item\_viii:w 112, 113, 4171,  
 4190, 4190  
 \\_\_enumext\_starred\_item\_viii\_aux\_i:w .. 113,  
 4190, 4195, 4198  
 \\_\_enumext\_starred\_item\_viii\_aux\_ii:w . 113,  
 4190, 4196, 4210, 4212  
 \\_\_enumext\_starred\_joined\_item\_vii:n 102, 108,  
 3646, 3646, 3950  
 \\_\_enumext\_starred\_joined\_item\_viii:n . 102,  
 112, 3646, 3695, 4169  
 \\_\_enumext\_starred\_ref: .... 39, 604, 642, 3192  
 \\_\_enumext\_starred\_ref:n .... 39, 598, 604, 604  
 \g\_\_enumext\_starred\_series\_tl . 60, 1605, 1606,  
 1775, 1778  
 \\_\_enumext\_start\_from:NNn 40, 701, 701, 714, 736  
 \l\_\_enumext\_start\_i\_int ..... 1717, 1729, 1748  
 \\_\_enumext\_start\_item\_tmp\_vii: 106, 3867, 3935,  
 3935  
 \\_\_enumext\_start\_item\_tmp\_viii: .. 111, 4111,  
 4154, 4154  
 \\_\_enumext\_start\_item\_vii:w 108, 109, 3963, 3968,  
 3993, 4000, 4002, 4002  
 \\_\_enumext\_start\_item\_viii:w .. 113, 4182, 4187,  
 4215, 4245, 4245  
 \g\_\_enumext\_start\_line\_tl 31, 34, 256, 271, 325,  
 1949, 1954, 1959, 1973, 1978, 1983  
 \\_\_enumext\_start\_list:nn .. 33, 89, 100, 343, 345,  
 3213, 3381, 3534, 3860, 4104  
 \\_\_enumext\_start\_mini\_vii: 106, 3744, 3744, 3904  
 \\_\_enumext\_start\_mini\_viii: ... 111, 3799, 3799,  
 4146  
 \\_\_enumext\_start\_save\_ans\_msg: 63, 1791, 1791,  
 1816  
 \\_\_enumext\_start\_store\_level: . 93, 3212, 3254,  
 3254  
 \\_\_enumext\_start\_store\_level\_vii: 107, 3859,  
 3915, 3915  
 \l\_\_enumext\_start\_vii\_int ... 1722, 1736, 1757  
 \l\_\_enumext\_start\_X\_int ..... 96, 731  
 \\_\_enumext\_stop\_item\_tmp\_vii: .. 106, 107, 109,  
 3866, 3870, 3937, 4004  
 \\_\_enumext\_stop\_item\_tmp\_viii: 111, 112, 4110,  
 4114, 4156, 4247  
 \\_\_enumext\_stop\_item\_vii: 109, 110, 4004, 4059,  
 4059  
 \\_\_enumext\_stop\_item\_viii: 115, 4247, 4296, 4296  
 \\_\_enumext\_stop\_list: .. 33, 343, 346, 3222, 3391,  
 3547, 3872, 4117  
 \\_\_enumext\_stop\_mini\_vii: 104, 107, 3744, 3763,  
 3908  
 \\_\_enumext\_stop\_mini\_viii: 112, 3799, 3818, 4150  
 \\_\_enumext\_stop\_save\_ans\_msg: . 63, 1791, 1796,  
 2008  
 \\_\_enumext\_stop\_store\_level: .. 93, 3223, 3254,  
 3283  
 \\_\_enumext\_stop\_store\_level\_vii: . 107, 3873,  
 3915, 3925  
 \l\_\_enumext\_store\_active\_bool . 28, 63, 93, 106,  
 108, 1746, 1755, 1823, 2293, 3258, 3271, 3396, 3404,  
 3492, 3551, 3917, 3927, 4129  
 \\_\_enumext\_store\_active\_keys:n 69, 2079, 2079,  
 3251  
 \\_\_enumext\_store\_active\_keys\_vii:n . 69, 106,  
 2079, 2089, 3895  
 \\_\_enumext\_store\_addto\_prop:n 71, 82, 2154, 2154,  
 2162, 2332, 2750, 4220  
 \\_\_enumext\_store\_addto\_seq:n 71, 83, 2163, 2163,  
 2167, 2174, 2188, 2196, 2205, 2219, 2227, 2390, 2843  
 \l\_\_enumext\_store\_anskey\_arg\_tl 28, 74, 75, 108,  
 2338, 2343, 2345, 2350, 2357, 2360, 2370, 2375, 2378,  
 2384, 2390  
 \\_\_enumext\_store\_anskey\_code:n 73, 74, 81, 2287,  
 2330, 2330, 2707, 2715  
 \l\_\_enumext\_store\_anskey\_env\_tl 28, 80, 81, 108,  
 2645, 2647, 2702, 2709, 2717  
 \l\_\_enumext\_store\_anskey\_opt\_tl 28, 80, 81, 108,  
 2646, 2663, 2669, 2676, 2682, 2692, 2704, 2713  
 \\_\_enumext\_store\_anskey\_safe\_outer: .... 73  
 \g\_\_enumext\_store\_columns\_break\_bool . 2570,  
 2660, 2725  
 \l\_\_enumext\_store\_columns\_break\_bool . 2246,  
 2340  
 \l\_\_enumext\_store\_current\_label\_tl 28, 82, 83,  
 113, 108, 2733, 2736, 2739, 2746, 2748, 2750, 2807,  
 2810, 2813, 2819, 2824, 2834, 2843, 4200, 4205, 4206,  
 4219, 4220, 4222  
 \l\_\_enumext\_store\_current\_label\_tmp\_tl . 28,  
 108, 3036, 3040  
 \l\_\_enumext\_store\_current\_opt\_arg\_tl 28, 113,  
 108, 2853, 2866, 2872, 4208  
 \\_\_enumext\_store\_internal\_ref: .. 74, 75, 2335,  
 2392, 2392  
 \g\_\_enumext\_store\_item\_join\_int .. 2573, 2667,  
 2671, 2726  
 \l\_\_enumext\_store\_item\_join\_int .. 2249, 2348,  
 2352  
 \g\_\_enumext\_store\_item\_star\_bool . 2575, 2674,  
 2727  
 \l\_\_enumext\_store\_item\_star\_bool . 2251, 2355  
 \g\_\_enumext\_store\_item\_symbol\_sep\_dim 2580,  
 2689, 2694, 2729  
 \l\_\_enumext\_store\_item\_symbol\_sep\_dim 2256,  
 2367, 2372  
 \g\_\_enumext\_store\_item\_symbol\_tl . 2578, 2680,  
 2684, 2728  
 \l\_\_enumext\_store\_item\_symbol\_tl . 2254, 2358,  
 2362  
 \l\_\_enumext\_store\_keyans\_item\_opt\_sep\_  
 tl .... 2032, 2744, 2746, 2817, 2821, 4203, 4205

\\_\_enumext\_store\_level\_close: . 71, 2168, 2192, 3287  
 \\_\_enumext\_store\_level\_close\_vii: . 72, 2199, 2223, 3931  
 \\_\_enumext\_store\_level\_open: . . 71, 2168, 2168, 3266, 3279  
 \\_\_enumext\_store\_level\_open\_vii: . . 72, 2199, 2199, 3921  
 \g\_\_enumext\_store\_name\_tl . 28, 63, 95, 108, 324, 331, 332, 333, 334, 1799, 1825, 1948, 1953, 1958, 1972, 1977, 1982, 2006  
 \l\_\_enumext\_store\_name\_tl 28, 63, 65, 108, 1632, 1635, 1659, 1662, 1750, 1759, 1794, 1803, 1804, 1825, 1826, 1827, 1829, 1830, 1832, 1834, 1835, 1837, 1839, 1840, 1864, 2156, 2158, 2165, 2458, 2459, 2495, 2649, 2790, 2791, 2894, 2907, 4237  
 \l\_\_enumext\_store\_ref\_key\_bool 74, 2055, 2333, 2381, 2754, 2831  
 \l\_\_enumext\_store\_save\_key\_vii\_bool . . 2091, 2121  
 \l\_\_enumext\_store\_save\_key\_vii\_tl 2093, 2094, 2122, 2123, 2203, 2211, 2215, 2219  
 \l\_\_enumext\_store\_save\_key\_X\_bool . . 69, 122  
 \l\_\_enumext\_store\_save\_key\_X\_tl . . . . 69, 122  
 \l\_\_enumext\_store\_upper\_level\_X\_bool . . 122  
 \\_\_enumext\_storing\_exec: 63, 78, 1801, 1817, 1821  
 \\_\_enumext\_storing\_set:n . . 63, 1786, 1801, 1801  
 \l\_\_enumext\_the\_counter\_v\_tl . . . . . 686  
 \l\_\_enumext\_the\_counter\_vii\_tl . . . . . 616  
 \l\_\_enumext\_the\_counter\_viii\_tl . . . . . 633  
 \l\_\_enumext\_the\_counter\_X\_tl . . . . . 49  
 \\_\_enumext\_tmp:n 44, 48, 53, 59, 70, 77, 78, 83, 90, 95, 96, 107, 125, 132, 151, 155, 159, 178, 795, 804, 850, 854, 1534, 1545, 1782, 1790, 1843, 1861, 2022, 2060, 2061, 2078, 2097, 2110, 2394, 2401, 2402, 2423, 2437, 2440, 2452, 2756, 2763, 3134, 3173, 3174, 3206  
 \\_\_enumext\_tmp:nn 469, 490, 491, 522, 523, 538, 731, 750, 831, 849, 907, 915, 916, 930, 995, 1011, 1012, 1025, 1423, 1439, 2919, 2934  
 \\_\_enumext\_tmp:nnn 539, 555, 556, 557, 558, 586, 602, 603  
 \\_\_enumext\_tmp:nnnnn 751, 776, 779, 782, 784, 786, 789, 792  
 \\_\_enumext\_tmp:w . . . . . 4348, 4351  
 \l\_\_enumext\_tmpa\_vii\_int . . . . . 3618, 3621  
 \l\_\_enumext\_tmpa\_viii\_int . . . . . 3637, 3640  
 \l\_\_enumext\_tmpa\_X\_int . . . . . 159  
 \l\_\_enumext\_topsep\_v\_skip 1083, 1087, 1230, 1243, 1251, 1256, 1276, 1280, 3550, 3582  
 \l\_\_enumext\_topsep\_vii\_skip . . 1307, 1316, 1320  
 \l\_\_enumext\_topsep\_viii\_skip . 1329, 1351, 1355  
 \\_\_enumext\_undefine\_anskey\_env: . 67, 77, 2017, 2503, 2503  
 \l\_\_enumext\_vspace\_a\_star\_v\_bool . . . . 1472  
 \l\_\_enumext\_vspace\_a\_star\_vii\_bool . . . 1494  
 \l\_\_enumext\_vspace\_a\_star\_viii\_bool . . . 1505  
 \l\_\_enumext\_vspace\_a\_star\_X\_bool . . . . . 96  
 \\_\_enumext\_vspace\_above: . . 56, 1440, 1440, 3292  
 \\_\_enumext\_vspace\_above\_v: . 56, 1468, 1468, 3420  
 \l\_\_enumext\_vspace\_above\_v\_skip . . 1470, 1474, 1476  
 \\_\_enumext\_vspace\_above\_vii: . . 57, 1490, 1490, 3901  
 \l\_\_enumext\_vspace\_above\_vii\_skip 1492, 1496, 1498  
 \\_\_enumext\_vspace\_above\_viii: . 57, 1490, 1501, 4144  
 \l\_\_enumext\_vspace\_above\_viii\_skip 1503, 1507, 1509  
 \l\_\_enumext\_vspace\_b\_star\_v\_bool . . . . 1483  
 \l\_\_enumext\_vspace\_b\_star\_vii\_bool . . . 1516  
 \l\_\_enumext\_vspace\_b\_star\_viii\_bool . . . 1527  
 \l\_\_enumext\_vspace\_b\_star\_X\_bool . . . . . 96  
 \\_\_enumext\_vspace\_below: . . 56, 1454, 1454, 3371  
 \\_\_enumext\_vspace\_below\_v: . 56, 1479, 1479, 3488  
 \l\_\_enumext\_vspace\_below\_v\_skip . . 1481, 1485, 1487  
 \\_\_enumext\_vspace\_below\_vii: . . 57, 1512, 1512, 3911  
 \l\_\_enumext\_vspace\_below\_vii\_skip 1514, 1518, 1520  
 \\_\_enumext\_vspace\_below\_viii: . 57, 1512, 1523, 4152  
 \l\_\_enumext\_vspace\_below\_viii\_skip 1525, 1529, 1531  
 \\_\_enumext\_widest\_from:nnn . . 41, 715, 715, 730, 742  
 \g\_\_enumext\_widest\_label\_tl 27, 36, 66, 457, 461, 465  
 \l\_\_enumext\_wrap\_label\_opt\_v\_bool . . . . 3030  
 \l\_\_enumext\_wrap\_label\_opt\_vii\_bool 108, 3962  
 \l\_\_enumext\_wrap\_label\_opt\_viii\_bool . . 113, 4181  
 \l\_\_enumext\_wrap\_label\_opt\_X\_bool . . . . . 96  
 \l\_\_enumext\_wrap\_label\_v\_bool 3026, 3030, 3038, 3094  
 \l\_\_enumext\_wrap\_label\_vii\_bool . . 108, 3961, 3966, 3974, 4043  
 \l\_\_enumext\_wrap\_label\_viii\_bool . 113, 4180, 4185, 4193, 4272  
 \l\_\_enumext\_wrap\_label\_X\_bool . . . . . 96  
 \\_\_enumext\_wrapper\_label\_v:n . . . . . 3096, 3529  
 \\_\_enumext\_wrapper\_label\_vii:n . . . . . 4046  
 \\_\_enumext\_wrapper\_label\_viii:n . . . . . 4275  
 \l\_\_enumext\_write\_aux\_file\_tl . 29, 76, 83, 148, 2461, 2467, 2797, 2803  
 \\_\_enumext\_zero\_parsep: . . 51, 1127, 1182, 1182  
 enumext\* . . . . . 5, 3853  
 enumXi . . . . . 429  
 enumXii . . . . . 429  
 enumXiii . . . . . 429  
 enumXiv . . . . . 429  
 enumXv . . . . . 429  
 enumXvi . . . . . 429  
 enumXvii . . . . . 429  
 enumXviii . . . . . 429  
 Environments provide by enumext:  
 anskey\* . . . . . 28, 63, 67, 75–78, 80, 93, 117, 119, 122  
 enumext\* 25, 26, 29–31, 33, 35, 38, 39, 41–44, 46, 47, 53, 54, 57–60, 62–65, 68–78, 80, 82, 85, 91–93, 105–107, 109, 111, 112, 114, 116, 117, 120, 123  
 enumext 25, 26, 30, 31, 33, 35–45, 47–60, 62–65, 68–78, 80, 82, 85–90, 92, 93, 96, 100, 101, 104, 107, 116, 117, 120, 122  
 keyans\* 25, 26, 28–30, 32, 35, 38–41, 43, 44, 46, 47, 53, 54, 57, 63, 64, 67, 68, 71, 78, 82, 85, 91, 111, 120, 123  
 keyanspic 25, 26, 28, 32, 35, 36, 40, 54, 63, 64, 67, 71, 78, 82–84, 98–101, 123  
 keyans . . 25, 26, 28, 30, 32, 35, 36, 40, 41, 43–47, 49, 52,

54–56, 63, 64, 67, 68, 71, 78, 82–84, 88–90, 96, 98–100, 104, 112, 120, 123

Environments:

list ..... 30, 33, 89, 90, 92

lrbox ..... 101, 109, 110, 114, 115

minipage ... 30, 33, 47, 49, 98, 100, 101, 109, 110, 115

multicols ..... 47–50, 54, 94, 95, 97, 98

scontents ..... 78, 80

exp commands:

\exp\_after:wN ..... 4351

\exp\_args:Ne ..... 2706, 2714, 3248, 4339

\exp\_args:NV ..... 2263, 2590

\exp\_not:N . 57, 460, 574, 619, 636, 689, 864, 878, 879, 890, 891, 902, 903, 2386, 2492, 2493, 2836, 2891, 2892, 2904, 2905, 4234, 4235, 4348

\exp\_not:n 258, 273, 285, 292, 299, 513, 533, 574, 575, 619, 620, 636, 637, 689, 690, 865, 1563, 1575, 2043, 2140, 2152, 2352, 2362, 2372, 2386, 2387, 2464, 2671, 2684, 2694, 2800, 2838, 2840

F

\fbox ..... 2027

file commands:

\file\_input\_stop: ..... 4800

first ..... 916

font ..... 469

\footnote ..... 85

\footnote ..... 85, 2943

\footnotemark ..... 2953

\footnotesize ..... 2493, 2892, 2905, 4235

\footnotetext ..... 2937

G

\getkeyans ..... 16, 116, 4337

group commands:

\group\_begin: .. 2278, 2491, 2625, 2701, 2890, 2903, 4022, 4041, 4233, 4260, 4270, 4359, 4393

\group\_end: 2289, 2498, 2721, 2897, 2910, 4051, 4063, 4240, 4280, 4300, 4361, 4400

H

\hbadness ..... 4070, 4307

hbox commands:

\hbox\_set:Nn ..... 449

\hfill 499, 503, 508, 509, 1394, 1412, 2386, 2836, 3768, 3823

hook commands:

\hook\_gput\_code:nnn ..... 9, 186, 190, 194, 367

\hook\_gremove\_code:nn ..... 80, 2641

\hook\_gset\_rule:nnnn ..... 368

\hook\_if\_empty:nTF ..... 2639

\hspace ..... 4081, 4319

\hyperlink ..... 75, 83

\hyperlink ..... 2386, 2836

\hypertarget ..... 34

\hypertarget ..... 397

I

\IfHyperBoolean ..... 375

\IfPackageLoadedTF ..... 11, 19, 371, 385

\ignorespaces ..... 867

\inputlineno ..... 258, 273, 285, 292, 299

int commands:

\int\_add:Nn ..... 3679, 3728

\int\_case:nn ... 1040, 1184, 1874, 1900, 1939, 1963

\int\_compare:nNnTF .. 356, 607, 624, 644, 651, 1109, 1228, 1373, 1377, 1381, 1987, 1993, 2004, 2297, 2301,

2305, 2317, 2536, 2540, 2544, 2734, 2773, 2778, 2783, 2808, 2886, 3231, 3241, 3263, 3276, 3315, 3331, 3345, 3359, 3405, 3409, 3438, 3463, 3476, 3496, 3500, 3556, 3649, 3659, 3675, 3698, 3708, 3724, 3881, 3919, 3929, 4076, 4085, 4123, 4130, 4313, 4323, 4470

\int\_compare\_p:nNn ... 228, 237, 249, 250, 264, 265, 1880, 1906, 2348, 2407, 2417, 2429, 2430, 2445, 2447, 2513, 2514, 2525, 2526, 2667, 3273

\int\_decr:N ..... 3678, 3727

\int\_eval:n . 341, 2158, 2459, 2493, 2791, 2892, 2905, 3149, 3191, 3667, 3716, 4235

\int\_from\_alph:n ..... 709, 723

\int\_from\_roman:n ..... 711, 725

\int\_gadd:Nn ..... 3680, 3729

\int\_gdecr:N ..... 1883, 1888, 1892, 1896, 1909

\int\_gincr:N 1716, 1721, 2286, 2846, 2974, 3005, 3044, 3305, 3429, 3518, 3939, 4017, 4158, 4224

\int\_gset:Nn ..... 1932, 2951

\int\_gset\_eq:NN 1615, 1622, 1628, 1634, 1642, 1649, 1655, 1661, 2948

\int\_gzero:N . 312, 313, 314, 1402, 1419, 1999, 2726, 3364, 3481, 4094, 4334

\int\_if\_exist:NNTF 1590, 1626, 1632, 1653, 1659, 1837

\int\_incr:N 2316, 3230, 3400, 3555, 3880, 3938, 4122, 4157

\int\_mod:nn ..... 4087, 4325

\int\_new:N . 28, 29, 30, 31, 32, 33, 60, 61, 84, 100, 119, 135, 136, 141, 142, 143, 145, 156, 162, 163, 164, 165, 166, 1592, 1840

\int\_set:Nn 705, 709, 711, 1729, 1736, 1748, 1757, 2626, 3595, 3596, 3618, 3637, 3648, 3654, 3670, 3697, 3703, 3719, 4070, 4307, 4466

\int\_set\_eq:NN ..... 1717, 1722, 3677, 3726

\int\_sign:n ..... 1934

\int\_step\_function:nnN ..... 2423, 2437, 2452

\int\_step\_inline:nnn ..... 3597

\int\_to\_roman:n ..... 198, 2403, 2441

\int\_use:N 334, 339, 340, 1110, 1731, 1738, 1750, 1759, 3149, 3168, 3191, 3249, 3316, 3325, 3340, 3346, 3652, 3653, 3665, 3701, 3702, 3714

\int\_zero:N ..... 4079, 4317

\item . 86, 87, 107, 109, 112, 114, 347, 2176, 2182, 2207, 2213, 2345, 2810, 2813, 3013, 3048, 3865, 3867, 4109, 4111, 4222

\item\* ..... 5, 14, 67, 3046

item-pos\* ..... 2919

item-sym\* ..... 2919

\itemindent ..... 90

\itemindent ..... 90

itemindent ..... 831

\itemsep ..... 100, 101

\itemsep ..... 3571, 3577

\itemwidth ..... 3625, 3644, 3688, 3692, 3737, 3741

K

keyans ..... 14, 3376

keyans\* ..... 14, 4098

keyanspic ..... 15, 3531

Keys for command provide by enumext:

break-col ..... 72, 74, 79, 80

item-join ..... 72, 74, 79, 80

item-pos\* ..... 72, 75, 79, 81

item-star ..... 72, 75, 79, 81

item-sym\* ..... 72, 75, 79, 81



Keys for environments provide by **enumext**:

above*	27, 55–57
above	27, 55–57, 94, 97, 106, 111
after	44–46, 95, 98, 107, 112
align	27, 37, 88, 109, 119
base-fix	42, 58, 70, 117
before*	44–46, 94, 106, 111
before	44–46, 97
below*	27, 55–57
below	27, 55–57, 95, 98, 107, 112
check-ans	28, 30, 31, 62–64, 66, 67, 70, 84, 86, 87, 94–96, 111, 121
columns-sep	47, 94, 97
columns	27, 47, 49, 55, 94, 97
first	44–46, 109
font	36, 88, 109
item-pos*	85, 87
item-sym*	28, 85, 87
itemindent	27, 43, 88, 109
itemsep	41, 91
labelsep	36, 86, 90, 109
labelwidth	35–38, 40, 41, 90
label	26, 27, 35–37, 40, 41, 101
lisparindent	91
list-indent	27, 43, 100
list-offset	43
listparindent	43, 109
mark-ans	68, 70, 77
mark-pos	68, 119
mark-ref	68, 70, 75
mini-env	27, 33, 47, 54, 55, 70, 85, 94, 97, 104–106, 111
mini-right*	27, 30, 47, 70, 104, 105
mini-right	27, 30, 47, 54, 70, 104, 105
mini-sep	27, 47, 70, 94, 97
no-store	28, 62–65, 70, 73
noitemsep	41, 51
nosep	41, 51
parindent	91
parsep	41, 91, 109
partopsep	41
ref	26, 30, 37, 38, 40, 121
resume*	26, 57, 58, 62, 63, 70, 95
resume	26, 32, 57–63, 70, 95, 107
rightmargin	43
save-ans	28, 32, 58–63, 65–67, 69–71, 73, 77, 78, 82, 83, 87, 93, 96, 99, 107, 112, 113, 116, 121
save-key	28, 58, 69, 93
save-pos	70
save-ref	29, 34, 68, 70, 74, 75, 82, 83, 88, 113
save-sep	68, 70, 113
series	26, 57–62, 70, 93, 95
show-ans	68, 70, 72, 74, 77, 88, 113
show-length	30, 44, 120
show-pos	28, 68, 72, 74, 77, 84, 88, 113
start	27, 30, 40, 41, 58
store-key	69
topsep	41
widest	27, 30, 41
wrap-ans	68, 70, 72, 77
wrap-label*	36, 86, 88, 108, 109, 113
wrap-label	36, 88, 108, 109, 113
wrap-opt	68, 70

keys commands:

\keys_define:nn	471, 493, 525, 541, 588, 659, 733, 753, 797, 833, 852, 909, 918, 997, 1014, 1425, 1536, 1784,
-----------------	---

1845, 2024, 2063, 2099, 2104, 2244, 2568, 2604, 2921, 4364, 4435	
\l_keys_key_str	73, 79, 2263, 2590, 4572
\keys_precompile:nnN	117, 4363, 4366, 4370, 4374, 4378, 4382, 4386
\keys_set:nn	485, 813, 824, 1020, 1430, 1435, 1678, 1683, 1770, 1778, 2284, 3243, 3248, 3416, 3893, 4139, 4418, 4425, 4437, 4438, 4439, 4440, 4441, 4442, 4443, 4444, 4445, 4446, 4447, 4448, 4449, 4487
\keys_set_known:nn	2713
keyval commands:	
\keyval_parse:NNn	1550, 2129

L

label	539, 586, 659
Labels provide by <b>enumext</b> :	
\Alph*	35, 36
\Roman*	35, 36
\alph*	35, 36
\arabic*	30, 35, 36
\roman*	35, 36
\labelsep	101
\labelsep	3572, 3575
labelsep	469
\labelwidth	36, 101
\labelwidth	3572, 3573
labelwidth	469
\leftmargin	90
\leftmargin	90, 3572
legacy commands:	
\legacy_if:nTF	4005, 4008, 4248, 4251
\legacy_if_gset_false:n	361
\legacy_if_set_false:n	4007, 4250
\legacy_if_set_true:n	3967, 3992, 3999, 4012, 4186, 4214, 4255
\linewidth	94, 97
\linewidth	3300, 3426, 3594, 3621, 3640, 3750, 3805
\list	345
list-indent	831
list-offset	831
\listparindent	3574
listparindent	831
\lrbox	4023, 4261

M

\makebox	101
\makebox	2234, 2236, 3068, 4037, 4045, 4049, 4274, 4278
\makelabel	86, 88, 89, 101
\makelabel	88, 89, 3074, 3090
\makesavenoteenv	391
mark-ans	2022
mark-pos	2022, 2061
mark-ref	2022
mini-env	995
mini-sep	995
\minipage	351
\miniright	10, 54, 1371, 3362, 3479
mode commands:	
\mode_if_math:TF	2325, 2552
\mode_if_vertical:TF	1065, 1093, 1209, 1288
\mode_leave_vertical:	811, 822, 864, 878, 890, 902, 2232, 3066, 4035
msg commands:	
\msg_error:nn	2314, 2319, 2323, 2550, 3407, 3411, 3498, 3558, 3883, 4125, 4132, 4450

\msg_error:nnn	564, 611, 628, 681, 1375, 1379, 1404, 1421, 1690, 1694, 1809, 2269, 2327, 2518, 2530, 2538, 2542, 2546, 2554, 2596, 4353, 4358, 4432, 4503
\msg_error:nnnn	2272, 2295, 2299, 2303, 2307, 2599, 3398, 3494, 3502, 4413
\msg_error:nnnnn	512, 532, 2042
\msg_fatal:nn	3232
\msg_fatal:nnn	423
\msg_info:nnn	13, 16, 21, 24, 373, 387
\msg_line_context:	4537, 4542, 4547, 4576, 4581, 4586, 4601, 4616, 4620, 4624, 4628, 4632, 4636, 4643, 4650, 4656, 4670, 4674, 4679, 4683, 4687, 4691, 4696, 4700, 4704, 4708, 4713, 4748, 4752, 4757, 4762, 4766, 4771, 4775, 4780, 4785, 4790, 4794, 4798
\msg_log:nnn	1829, 1834, 1839
\msg_log:nnnnn	338, 1972, 1977, 1982
\msg_log:nnnnnn	330
\msg_new:nnn	4504, 4508, 4512, 4516, 4521, 4534, 4539, 4544, 4549, 4558, 4566, 4570, 4574, 4579, 4584, 4599, 4614, 4618, 4622, 4626, 4630, 4634, 4638, 4647, 4653, 4659, 4663, 4667, 4672, 4677, 4681, 4685, 4689, 4694, 4698, 4702, 4706, 4711, 4746, 4750, 4755, 4760, 4764, 4769, 4773, 4778, 4783, 4788, 4792, 4796
\msg_new:nnnn	4525, 4716, 4725, 4734, 4740
\msg_term:nnnn	1793, 1798, 3158, 3168, 3197, 3202
\msg_term:nnnnn	1953
\msg_warning:nn	3361, 3478
\msg_warning:nnnn	1990, 1996, 3106, 3111, 3651, 3664, 3700, 3713
\msg_warning:nnnnn	1948, 1958
\multicolsep	94, 97
\multicolsep	3330, 3451

N

\NeedsTeXFormat	3
\newcounter	426
\NewDocumentCommand	1371, 2275, 3490, 4337, 4391, 4457
\NewDocumentEnvironment	3207, 3376, 3531, 3853, 4098
\newenvsc	2561
\newlabel	35
\newlabel	409
no-store	1843
\noindent	106, 111
\noindent	3307, 3431, 3759, 3814, 3866, 4078, 4110, 4316
\nointerlineskip	3307, 3431, 3759, 3814
noitemsep	751
\nopagebreak	1076, 1104, 1220, 1299, 1362, 1368
\normalfont	2492, 2891, 2904, 4234
nosep	751

P

Packages:	
caption	104
enumext	25, 37, 62, 90, 99, 119
enumitem	35
expl3	101
footnotehyper	34
hyperref	29, 30, 34, 35, 75, 83, 109, 119
lua-visual-debug	49
multicol	25, 119
scontents	25, 77, 78
shortlst	101
\par	1076, 1104, 1220, 1299, 1362, 1368, 1397, 1414, 2471, 3351, 3366, 3468, 3483, 3606, 3777, 3791, 3832, 3846, 4078, 4092, 4316, 4332
\parindent	4055, 4284

\parsep	48, 51, 100, 101
\parsep	3188, 3571, 3578, 3583
parsep	751
\parskip	4056, 4285
\partopsep	101
\partopsep	3189, 3576
partopsep	751
peek commands:	
\peek_meaning:NTF	3944, 3958, 3975, 3986, 4163, 4177, 4194
\peek_meaning_remove:NTF	3951, 4170
\peek_remove_spaces:n	3052
\phantomsection	34
\phantomsection	398
prg commands:	
\prg_do_nothing:	402
\prg_new_protected_conditional:Npnn	200
\prg_replicate:nn	217
\prg_return_false:	204
\prg_return_true:	203
\printkeyans	16, 116, 4391
prop commands:	
\prop_count:N	332, 2158, 2459, 2495, 2791, 2894, 2907, 4237
\prop_gput_if_not_in:Nnn	2156
\prop_if_exist:NTF	1827, 4357
\prop_item:Nn	4360
\prop_new:N	1830
\ProvidesExplPackage	4

R

\raggedcolumns	3339, 3457
\ref	75, 82
ref	539, 586, 659
\refstepcounter	4014, 4257
regex commands:	
\regex_match:nnTF	202, 708, 710, 722, 724
\regex_replace_once:nnN	210
\renewcommand	574, 619, 636, 689
\RenewDocumentCommand	2943, 3013, 3048, 3074, 3090
\RequirePackage	17, 25
resume	1534
resume*	1534
rightmargin	831
\Roman	36, 40, 41
\Roman	445
\roman	36, 40, 41
\roman	446, 557, 4381

S

save-ans	1782
save-key	2097
save-ref	2022
save-sep	2022
scan commands:	
\scan_stop:	101, 3585, 3865, 4109, 4348, 4351
scontents internal commands:	
\l_scontents_fname_out_tl	2614
\__scontents_parse_environment_keys:n	2620
\__scontents_rescan_tokens:n	2627
\l_scontents_storing_bool	2612
\l_scontents_writing_bool	2613
seq commands:	
\seq_clear:N	4464
\seq_const_from_clist:Nn	4452

©2024 by Pablo González L

<code>\tl_set:Nn</code> .	57, 283, 290, 297, 425, 499, 503, 508, 509, 561, 606, 678, 862, 876, 888, 900, 1702, 1803, 2084, 2094, 2115, 2123, 2489, 2614, 2853, 2888, 2901, 2991, 4208, 4231, 4492
<code>\tl_set_eq:NN</code>	466, 567, 570, 614, 616, 631, 633, 684, 686, 2396, 2758, 2771, 3036, 3040, 3523, 3525
<code>\tl_to_str:n</code> . . . . .	1673, 1679, 1684, 4340
<code>\tl_trim_spaces:n</code> . . . . .	456, 4480, 4492, 4498
<code>\tl_use:N</code> .	462, 465, 583, 648, 655, 698, 933, 937, 941, 945, 949, 953, 957, 961, 965, 969, 973, 977, 981, 985, 989, 993, 2238, 2403, 2411, 2422, 2436, 2441, 2453, 2978, 2984, 3009, 3027, 3031, 3039, 3076, 3077, 3084, 3092, 3093, 3099, 3214, 3382, 3528, 3787, 3842, 4042, 4053, 4057, 4271, 4282, 4288, 4293, 4394, 4395, 4396, 4397, 4398, 4416, 4476
token commands:	
<code>\token_to_str:N</code> . . . . .	409
<code>topsep</code> . . . . .	<u>751</u>
<code>\typeout</code> . . . . .	377, 380, 390, 391
<b>U</b>	
<code>\u</code> . . . . .	211
use commands:	
<code>\use:N</code> . . . . .	218, 3081, 3216
<code>\use:n</code> . . . . .	1548, 2127, 4346
<code>\use_none:nn</code> . . . . .	401
<code>\usecounter</code> . . . . .	3148, 3190
<b>V</b>	
<code>\value</code> . . . .	1616, 1622, 1629, 1635, 1643, 1649, 1656, 1662
vbox commands:	
<code>\vbox_set_top:Nn</code> . . . . .	3785, 3840
<code>\vspace</code> . .	362, 812, 823, 1447, 1450, 1461, 1464, 1474, 1476, 1485, 1487, 1496, 1498, 1507, 1509, 1518, 1520, 1529, 1531, 3539, 3550, 4093, 4333
<b>W</b>	
<code>widest</code> . . . . .	<u>731</u>
<code>wrap-ans</code> . . . . .	<u>2022</u>
<code>wrap-label</code> . . . . .	<u>469</u>
<code>wrap-label*</code> . . . . .	<u>469</u>
<code>wrap-opt</code> . . . . .	<u>2022</u>