

enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-09-18*

©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

1	Introduction	1	6	The storage system	11
1.1	Description and usage	2	6.1	Keys for storage system	11
1.2	The concept of left margin	3	6.1.1	Keys for label and ref	11
1.3	User interface	3	6.1.2	Keys for wrap and display	12
1.3.1	Internal counters	3	6.1.3	Keys for debug and checking	12
1.3.2	Public dimension	3	6.2	The command <code>\anskey</code>	12
1.3.3	Support for <code>multicol</code>	3	6.2.1	Keys for <code>\anskey</code>	12
1.3.4	Support for <code>minipage</code>	4	6.3	The environment <code>anskey*</code>	13
1.3.5	The <code>\label</code> and <code>\ref</code> system	4	6.3.1	Keys for <code>anskey*</code>	13
1.3.6	Support for <code>\footnote</code>	4	6.4	The environment <code>keyans</code>	14
2	The environments provided	4	6.4.1	The <code>\item*</code> in <code>keyans</code>	14
2.1	The environment <code>enumext</code>	4	6.5	The environment <code>keyanspic</code>	15
2.2	The environment <code>enumext*</code>	5	6.5.1	The command <code>\anspic</code>	15
2.3	The command <code>\item*</code>	5	6.6	Printing stored content	16
2.3.1	Keys for <code>\item*</code>	5	6.6.1	The command <code>\getkeyans</code>	16
2.4	The command <code>\item</code> in <code>enumext*</code>	5	6.6.2	The command <code>\foreachkeyans</code>	16
3	The command <code>\setenumext</code>	6	6.6.3	The command <code>\printkeyans</code>	16
4	The command <code>\setenumextmeta</code>	6	7	Full examples	17
5	The <code>keyval</code> system	6	8	The way of non-enumerated lists	20
5.1	Keys for label and ref	6	9	References	22
5.2	Keys for spaces	7	10	Change history	22
5.2.1	Vertical spaces	7	11	Index of Documentation	23
5.2.2	Horizontal spaces	8	12	Implementation	25
5.3	Keys for add code	9	13	Index of Implementation	136
5.4	Keys for start, series and resume	9			
5.5	Keys for <code>multicols</code>	10			
5.6	Keys for <code>minipage</code>	10			
5.6.1	The command <code>\miniright</code>	10			
5.6.2	The key <code>mini-right</code>	10			

Motivation and acknowledgments

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

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

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

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

*This file describes a documentation for v1.0, last revised 2024-09-18.

†E-mail: pablgonz@educarchile.cl.

License and Requirements

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

1 Introduction

In the L^AT_EX 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^AT_EXze 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^AT_EXze is cool?

*

Very True!

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

*

Yes

(b) Usually uses the package manager?

*

Yes, dnf

(c) Rate the following package and class

i. xsim-exam

*

doesn’t exist for now :(

ii. xsim

*

very good

iii. exsheets

*

obsolete

Or we are interested in referring to a specific question and its “answer”, for example:
The answer to 3.(b) is “Very True!” and the answer to 4.(c).ii is “very good”.
Or we are interested in printing all the “answers”:

1. $(x - 1)^2$

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

3. (a) False

(b) Very True!

4. (a) Yes
- * (b) Yes, dnf

* (c) i. doesn’t exist for now :(

* ii. very good

* iii. obsolete

*

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

1. First type of questions

A) value

B) correct

C) value

D) value

2. Second type of questions

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

II. $\alpha = \delta$

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

A) I only

B) II only

C) I and II only

D) I and III only

E) I, II, and III

★ 3. Third type of questions

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

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

A) value

B) value

C) value

D) value

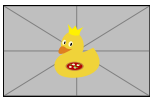
E) value
4. Question with image and label below:

A

A

B

B)



5. Question with image on left side:

B

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 / 150

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 \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 \TeX Live and \MiKTeX , use the package manager to install. For manual installation, download `enumext.zip` and unzip it, run `lualatex enumext.dtx` and move all files to appropriate locations, then run `mktexlsr`. To produce the documentation run `lualatex enumext.dtx` two times.

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

The package is loaded in the usual way:

```
\usepackage{enumext}
```

1.2 The concept of left margin

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

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



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

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

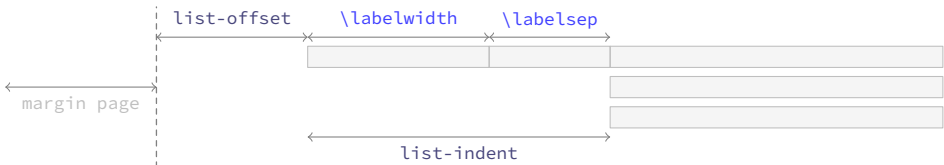


Figure 2: Representation of horizontal lengths concept in `enumext`.

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

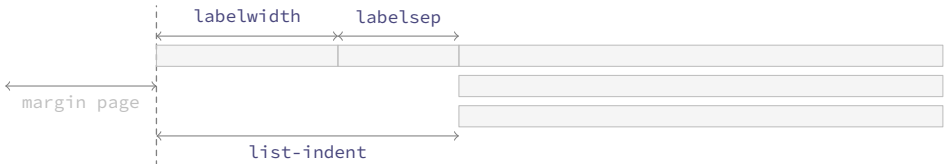


Figure 3: Default horizontal lengths `list-offset=0pt`, `list-indent=\labelwidth+\labelsep` in `enumext`.

1.3 User interface

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

1.3.1 Internal counters

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

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

1.3.2 Public dimension

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

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

1.3.3 Support for multicol

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



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

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

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

1.3.4 Support for minipage

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



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

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

1.3.5 The \label and \ref system

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

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

1.3.6 Support for \footnote

This package provides an internal implementation for the `\footnote` command which is compatible with the `hyperref` package for the `enumext*` and `keyans*` environments, but will not produce the expected links, and if the `mini-env` key is used in `enumext` or `keyans` environments the output will look like the classic way they are displayed in the environment `minipage`.
The best way to solve this is to use Jean-François Burnol `footnotehyper`[9] package, it will support keeping the links if `hyperref` is loaded with the `hyperfootnotes=true` option (default) and will show the output numbered at the bottom of the page (as opposed to how it is displayed in the `minipage` environment). The way to load it is as follows:

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

2 The environments provided

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

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

2.1 The environment enumext

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

Example with columns=2

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

2.2 The environment enumext*

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

Some considerations to take into account for this environment:

- The environment cannot be nested within itself or in the environment `keyans*`, but it can be nested within `enumext` and vice versa.
- Each “item” in the environment is placed within a `minipage` environment whose *width* is stored in the dimension `\itemwidth` that NOT includes `labelwidth`, `labelsep`, only the *width of the content*.
- You cannot have floating environments like `figure` or `table` but `\footnote` with `hyperref` support is supported if the `footnotehyper` package is loaded.

Example with columns=2

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

2.3 The command \item*

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

The `\item*`, `\item*[\langle symbol \rangle]` and `\item*[\langle symbol \rangle][\langle offset \rangle]` works like the numbered `\item`, but placing a `\langle symbol \rangle` to the “left” of the `\langle label \rangle` separated from it by the `\langle offset \rangle` set by the the second optional argument. The default values for `\langle symbol \rangle` and `\langle offset \rangle` are `\$star$ ‘★’` and the value set by `labelsep` key.
The *starred argument* ‘★’ cannot be separated by spaces ‘␣’ from the command, i.e. `\item*` and the first optional argument does “not support” verbatim content. Can be configure with the keys `item-sym*` and `item-pos*` locally in the environment or globally using `\setenumext` command (§3).

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

2.3.1 Keys for `\item*`

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

`item-pos*` = { $\langle rigid length \rangle$ } default: *by levels*
Sets the *offset* between the box containing the current $\langle label \rangle$ defined by `labelwidth` key and the $\langle symbol \rangle$ set by `item-sym*` key. The default values are set by `labelsep` key at each level. If positive values are passed it will *offset to the left* and if negative values are passed it will *offset to the right*.

2.4 The command `\item` in `enumext*`

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

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

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

3 The command `\setenumext`

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

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

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

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

4 The command `\setenumextmeta`

<code>\setenumextmeta</code>	<code>\setenumextmeta {$\langle key name \rangle$}{$\langle key-one = val, key-two = val, ... \rangle$}</code>
	<code>\setenumextmeta*{$\langle key name \rangle$}{$\langle key-one = val, key-two = val, ... \rangle$}</code>
	<code>\setenumextmeta [$\langle enumext* \rangle$] {$\langle key name \rangle$}{$\langle key-one = val, key-two = val, ... \rangle$}</code>
	<code>\setenumextmeta [$\langle enumext, level \rangle$] {$\langle key name \rangle$}{$\langle key-one = val, key-two = val, ... \rangle$}</code>

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

The starred version `*` will create the new “meta-key” for the environment `enumext*` and for all levels of the environment `enumext`.

5 The keyval system

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

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

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

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

5.1 Keys for label and ref

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

default: *empty*

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

5.2 Keys for spaces

`show-length = {⟨true | false⟩}`

default: *false*

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

5.2.1 Vertical spaces

`topsep = {⟨rubber length | rigid length⟩}`

default: *by levels*

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

`parsep = {⟨rubber length | rigid length⟩}`

default: *by levels*

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

`partopsep = {⟨rubber length | rigid length⟩}`

default: *by levels*

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

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

`itemsep = {⟨rubber length | rigid length⟩}`

default: *by levels*

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

`noitemsep` *⟨value forbidden⟩*

default: *not used*

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

`nosep` *⟨value forbidden⟩*

default: *not used*

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

`base-fix` *⟨value forbidden⟩*

default: *not used*

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

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

`above = {⟨rubber length | rigid length⟩}`

default: *not used*

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

`above* = {⟨rubber length | rigid length⟩}`

default: *not used*

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

`below = {⟨rubber length | rigid length⟩}`

default: *not used*

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

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

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

5.2.2 Horizontal spaces

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

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

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

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

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

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

`list-offset` = { $\langle rigid\ length \rangle$ } default: `0pt`

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

`list-indent` = { $\langle rigid\ length \rangle$ } default: `labelwidth + labelsep`

Sets the *indentation* of the whole environment under the box defined by `labelwidth` and `labelsep` keys. Internally sets the value of `\leftmargin` and `\itemindent` for the current level.

If `list-indent=0pt` is set in the environment `enumext` the $\langle label \rangle$ will be part of the text, separated by the value of the `labelsep` key and the *first word*, in simple terms it will look like a “*common paragraph*”. This setting is equivalent (more or less) to the `wide` key provided by the `enumitem` package.

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

5.3 Keys for add code

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

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

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

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

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

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

Executes $\{\langle code \rangle\}$ when “*starting*” the environment. The $\{\langle code \rangle\}$ must be passed between braces, is executed right “*after*” all *list parameters* are done, after the second argument of list, just before the first occurrence of `\item: \begin{list}{\langle arg one \rangle}{\langle arg two \rangle}{\langle code \rangle}\item`.

- Keep in mind that the code set in this key will affect the entire “*body*” of the environment and therefore the inner levels of the list and the `keyans` environment. It is recommended to set this key per level.

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

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

5.4 Keys for start, series and resume

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

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

`start*` = { $\langle integer \mid string \rangle$ } default: *not used*

Sets the *start value* of the numbering on the current level. Internally $\langle string \rangle$ is converted and passed as value to the counter defined by `label` key on the current level, i.e. it is equivalent to enter `start=5`, `start=E` or `start=v`.

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

`series = {⟨series name⟩}` default: *not used*

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

`resume = {⟨series name⟩}` default: *not used*

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

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

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

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

5.5 Keys for multicols

`columns = {⟨integer⟩}` default: `1`

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

`columns-sep = {⟨rigid length⟩}` default: *by level*

Set the *space between columns* used by the `multicols` environment within the environment. Internally sets the value of `\columnsep`, by default its value is equal to the sum of the values set in the keys `labelwidth` and `labelsep` of the current level.

- The `\footnote{⟨text⟩}` command in the nested levels of `multicols` will not work as expected, prefer the use of `\footnotemark[⟨number⟩]` inside the environment and `\footnotetext[⟨number⟩]{⟨text⟩}` outside the environment or via the `after` key.

5.6 Keys for minipage

`mini-env = {⟨rigid length⟩}` default: *not used*

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

`mini-sep = {⟨rigid length⟩}` default: `0.3333em`

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

5.6.1 The command `\miniright`

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

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

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

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

5.6.2 The key `mini-right`

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

`mini-right = {⟨content⟩}` default: *not used*

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

`mini-right* = {⟨content⟩}` default: *not used*

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

6 The storage system

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

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

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

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

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

6.1 Keys for storage system

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

`save-ans = {\store name}` default: *not set*

Sets the *name* of the $\langle sequence \rangle$ and $\langle prop list \rangle$ in which the contents will be “*stored*” by `\anskey` and `anskey*` in `enumext` and `enumext*` environments, `\item*` in `keyans` and `keyans*` environments and `\anspic*` in `keyanspic` environment. If the $\langle sequence \rangle$ or $\langle prop list \rangle$ does not exist, it will be created globally and will not be overwritten if the key is used again.

`save-key = {\key list}` default: *not set*

This key *overrides* the default “*stored keys*” of the optional arguments of the inner levels or nested environment that will be passed to the $\langle sequence \rangle$. The $\langle key list \rangle$ passed to this key ignores any $\langle keys \rangle$ in the “*stored system*” and must be passed between braces. For example, if we execute at a second level:

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

The $\langle keys \rangle$ that will be stored by default in the $\langle sequence \rangle$ would be `nosep`, `columns=2`, but using the key `save-key={columns=3}` will overwrite this and store it in the $\langle sequence \rangle$ only the key `columns=3` ignoring all the others.

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

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

6.1.1 Keys for label and ref

`save-ref = {\true | false}` default: *false*

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

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

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

6.1.2 Keys for wrap and display

- `wrap-ans` = {`<code> {#1} more code`} default: `\fbox+\parbox{#1}`
 Wraps the *argument* passed to the `\anskey` and the *body* in `anskey*` environment referenced by {#1} when using the `show-ans` or `show-pos` keys. The {`<code>`} must be passed between braces and only affects the *argument* or *body* and NOT the “stored content” in the *sequence* and *prop list* {`<store name>`} set by `save-ans` key. If this key is passed using `\setenumext` it is necessary to use double ‘{#1}’.
- `wrap-opt` = {`<code> {#1} more code`} default: `[{#1}]`
 Wraps the *optional argument* passed to the `\item*` and `\anspic*` referenced by {#1} in the `keyans`, `keyans*` and `keyanspic` environments when using the `show-ans` or `show-pos` keys. The {`<code>`} must be passed between braces and only affects the current *optional argument* and NOT the “stored content” in the *sequence* and *prop list* {`<store name>`} set by `save-ans` key. If this key is passed using `\setenumext` it is necessary to use double ‘{#1}’.
- `show-ans` = {`<true> | <false>`} default: `false`
 Displays the *argument* passed to the `\anskey`, the *body* for `anskey*` environment, the `<label>` for `\item*` and `\anspic*` at the place where it is executed. If the optional argument is present in `\item*` or `\anspic*` it will be shown using `wrap-opt` key.
- `mark-ans` = {`<symbol>`} default: `\textasteriskcentered`
 Sets the *symbol* to be displayed in the left margin for `\anskey`, `anskey*`, `\item*` and `\anspic*` in the place where they are executed when using the key `show-ans`.
- `mark-pos` = {`<left> | <right>`} default: `left`
 Sets the *aligned* of the symbol defined by `mark-ans` key. The “symbol” is aligned in a box with the same dimensions of the label box defined by `labelwidth` key on the current level and separated by the value of the `labelsep` key.

6.1.3 Keys for debug and checking

- `show-pos` = {`<true> | <false>`} default: `false`
 Displays the *position* occupied by the “stored content” by `\anskey`, `anskey*`, `\item*` and `\anspic*` in the *prop list* {`<store name>`} set by `save-ans` key. This position is used by the `\getkeyans` command and by the `\ref` command if the `save-ref` key is active.
- `check-ans` = {`<true> | <false>`} default: `false`
 Enables the *checking answer* mechanism displaying an appropriate message on the terminal. This key works under the logic that each `\item` or `\item*` that does not open an inner level or nested environment contains “only one answer” or “only one execution” of the `\anskey` or `anskey*`. It is intended to be used in conjunction with the `no-store` key.
- `no-store` `<value forbidden>` default: `not used`
 This is a *meta-key* that does not receive an argument and disables the structure stored in the *sequence* {`<store name>`} set by `save-ans` key at the entire level or a nested environment in which it runs. This key is intended for use in internal levels or nested `enumext` or `enumext*` environments in which you want to use `enumext` or `enumext*` but “without” using the `\anskey`, “without” use `anskey*`, “without” interfering with the `check-ans` key and “without” storing an unwanted structure in the *sequence* {`<store name>`}.

6.2 The command `\anskey`

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

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

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

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

6.2.1 Keys for `\anskey`

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

- `break-col` `<value forbidden>` default: `not used`
 Stores {`<content>`} in the *sequence* {`<store name>`} of the form `\columnbreak \item<content>`.
- `item-join` = {`<columns>`} default: `not set`
 Set the *number of columns* to be used for `\item(<columns>)` and stores {`<content>`} in the *sequence* {`<store name>`} of the form `\item(<columns>)<content>`.
- `item-star` `<value forbidden>` default: `not used`
 Stores {`<content>`} in the *sequence* {`<store name>`} of the form `\item*<content>`.

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

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

Example

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

- | | |
|--|---|
| * 1. Text containing our instructions or questions.
* <input type="text" value="first answer"/>
2. Text containing our instructions or questions.
(a) Question.
* <input type="text" value="second answer"/> | 3. Text containing our instructions or questions.
* <input type="text" value="third answer"/>
4. Text containing our instructions or questions.
* <input type="text" value="fourth answer"/> |
|--|---|

6.3 The environment `anskey*`

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

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

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

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

6.3.1 Keys for `anskey*`

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

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

`overwrite` = $\{\langle true | false \rangle\}$ default: *false*
 Sets whether the $\langle file.ext \rangle$ generated by `write-env` from the `anskey*` environment will be rewritten.

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

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

Example

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

```

\item Text containing our instructions or questions.
\begin{enumext}
  \item Question.
  \begin{anskey*}
    \langle second answer \rangle
  \end{anskey*}
\end{enumext}
\item Text containing our instructions or questions.
\begin{anskey*}
  \langle third answer \rangle
\end{anskey*}
\item Text containing our instructions or questions.
\begin{anskey*}
  \langle fourth answer \rangle
\end{anskey*}
\end{enumext}

```

- | | |
|---|--|
| <p>★ 5. Text containing our instructions or questions.</p> <p>[5] <input type="text" value="First answer with verbatim"/></p> <p>6. Text containing our instructions or questions.</p> <p>(a) Question.</p> <p>[6] <input type="text" value="second answer"/></p> | <p>7. Text containing our instructions or questions.</p> <p>[7] <input type="text" value="third answer"/></p> <p>8. Text containing our instructions or questions.</p> <p>[8] <input type="text" value="fourth answer"/></p> |
|---|--|

6.4 The environments `keyans` and `keyans*`

<p><code>keyans</code> <code>\begin{keyans}[\langle key = val \rangle] \item \item[\langle custom \rangle] \item* \item*[\langle content \rangle] \end{keyans}</code></p> <p><code>keyans*</code> <code>\begin{keyans*}[\langle key = val \rangle] \item \item[\langle custom \rangle] \item* \item*[\langle content \rangle] \end{keyans*}</code></p>
--

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

<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 `\keys` 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 `\keys` 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 `\label` which will be set to `label=\Alph*`.

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

<p><code>\item*</code> <code>\item*</code></p> <p><code>\item*</code> <code>\item*[\langle content \rangle]</code></p>
--

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

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

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

Example

```

\begin{enumext}[save-ans=test,columns=2,show-ans=true]
  \item Text containing a question.
  \begin{keyans*}[nosep,columns=2]
    \item Choice
    \item* Correct choice
    \item Choice
    \item Choice
  \end{keyans*}
\end{enumext}

```




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

1. Text containing a question.

A) Choice
C) Choice
E) Choice

* B) Correct choice
D) Choice
2. Text containing a question and image.

A) Choice
B) Choice
C) Choice
D) Choice
* E) [note] Correct choice


Some text

6.5 The environment keyanspic

keyanspic

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

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



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

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

6.5.1 The command \anspic

\anspic

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

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

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

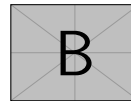
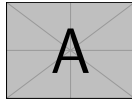
Example

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

1. Question with images.



A)



B)



C)

* E)[note]

6.6 Printing stored content

6.6.1 The command `\getkeyans`

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

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

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

6.6.2 The command `\foreachkeyans`

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

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

Options for command

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

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

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

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

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

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

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

6.6.3 The command `\printkeyans`

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

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

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

The default values for the “first level” are the same as the default values for the `enumext` and `enumext*` environments along with the keys `nosep`, `first=\small`, `font=\small` and `columns=2`. For the inner levels of the environment `enumext` saved in the *sequence* `{<store name>}` the default values are the same as those established for the second, third and fourth levels plus the keys `nosep`, `first=\small`, `font=\small`. If the environment `enumext*` is saved within the *sequence* `{<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 $\langle keys \rangle$ set for the first level unless they are set in the optional argument or `save-key` is used to modify it.
- If we execute `\printkeyans{⟨store name⟩}` and the *sequence* $\{⟨store name⟩\}$ contains any environment `enumext*`, they will start with the $\langle keys \rangle$ set by default unless they are set in the optional argument or `save-key` is used to modify it.

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

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]

$3(x + y + z)$

2. True False

(a)

~~LaTeX2e~~ is cool?

[2]

Very True!

3. Related to Linux

(a) You use linux?

[3]

Yes

(b) Rate the following package and class

i.

xsim

[4]

very good

ii.

exsheets

[5]

obsolete

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

1. $3(x + y + z)$

2. (a) Very True!

3. (a) Yes

(b) i. very good

ii. obsolete
- *

*

*

*

*

7 Full examples

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

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

and then you can use the excellent `arara`¹ tool to compile them.

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

- A

I only

B

II only

C

I and II only
- D

I and III only
- E

I, II, and III

3. Third type of questions

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

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

A

value

B

value

C

value

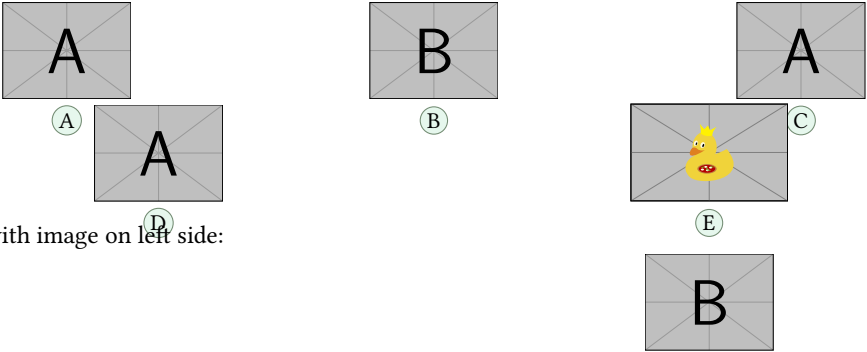
D

value

E

value

4. Question with image and label below:



5. Question with image on left side:
- A

value
- B

value
- C

value
- D

correct
- E

value

Test keys

1. B, $x = 5$

2. D

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

* 5. D, other note

*

Example 4

A “simple worksheet” using ducks :) 🦆

Factor $x^2 - 2x + 1$

Factor $3x + 3y + 3z$

The following questions need to be cuaqtified :)

True False

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

Related to Linux

- (a) You use linux?
- (b) Usually uses the package manager?
- (c) Rate the following package and class

i. `xsim-exam`

ii. `xsim`

iii. `exsheets`

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

1. $(x - 1)^2$

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

3. (a) False

(b) Very True!

4. (a) Yes
- (b) Yes, dnf

(c) i. doesn't exist for now :(

ii. very good

iii. obsolete

Example 5

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

1

Which choice best describes what happens in the passage?

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

B) One character receives a surprising request from another character.

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

D) One character criticizes another character for

2

pursuing an unexpected course of action.

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.

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)

8 The way of non-enumerated lists

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

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

Fake itemize environment

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

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

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

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

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 \LaTeX , that is to say the output does not always look as nice as it should, even if they are only alternatives these must follow a certain order when presented either numerical or presentation, that said handling that using *nested lists* is quite complicated so I do not classify to be implemented.

9 References

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

10 Change history

v1.0 2024-09-18 – First public release.

11 Index of Documentation

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

C

Document class:

article 2

book 2

exam 2

letter 2

report 2

\columnbreak 4, 12

\columnsep 10

Commands provide by enumext:

\anskey 11–13

\anspic 11, 12, 15

\foreachkeyans 16

\getkeyans 12, 16

\item* 5–7, 11, 12, 14, 15

\item 5–10, 12, 14

\miniright 10

\printkeyans 6, 11, 16

\setenumextmeta 6

\setenumext 5–7, 11, 12, 14, 17

Counters defined by enumext:

enumXiii 4

enumXii 4

enumXiv 4

enumXi 4

enumXviii 4

enumXvii 4

enumXvi 4

enumXv 4

E

Environments provide by enumext:

anskey* 11–13

enumext* 4–14, 16, 17

enumext 4–14, 16, 17, 20

keyans* 4–14

keyanspic 4, 7, 8, 11–13, 15, 20

keyans 4–9, 11–15, 20

Environments:

Verbatim 13

enumerate 1, 3, 5, 21

figure 5

list 3, 9, 21

minipage 3–5, 10, 21

multicols 3, 4, 10

table 5

task 5

F

\footnote 5

I

\itemsep 8

K

Keys for \anskey provide by enumext:

break-col 12

item-join 12

item-pos* 13

item-star 12, 13

item-sym* 13

Keys for \foreachkeyans provide by enumext:

after 16

before 16

sep 16

start 16

step 16

stop 16

wrapper 16

Keys for anskey* provide by enumext:

break-col 12

force-eol 13

item-join 12

item-pos* 13

item-star 12, 13

item-sym* 13

overwrite 13

write-env 13

Keys for environments provide by enumext:

above* 8

above 8

after 9, 10

align 7, 21

base-fix 8

before* 9

before 9

below* 9

below 8

check-ans 12

columns-sep 4, 10

columns 4, 8, 10

first 9

font 7

item-pos* 5, 6

item-sym* 5, 6

itemindent 9

itemsep 8, 15

labelsep 3–7, 9, 10, 12, 20, 21

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

labelwith 5

label 7, 9, 14, 20, 21

list-indent 3, 9

list-offset 3, 9, 21

listparindent 9

mark-ans 12

mark-pos 12

mark-ref 11

mini-env 4, 5, 8, 10

mini-right* 7, 10

mini-right 7, 10

mini-sep 4, 10

no-store 11–13

noitemsep 8

nosep 8, 20

overwrite 13

parsep 8, 15

partopsep 8

ref 4, 7

resume* 7, 10, 11

resume 7, 10, 11

rightmargin 9

save-ans 4, 6, 10–16

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

12 Implementation

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

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

12.1 General conventions

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

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

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

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

12.2 Initial set up

Start the DocStrip guards.

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

Identify the internal prefix (L^AT_EX3 DocStrip convention) for l3doc class.

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

12.3 Declaration of the package

First we will make sure we have a minimum (super updated) version of L^AT_EX to work correctly.

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

Now declare the `enumext` package.

```
4 \ProvidesExplPackage
5   {enumext}
6   {2024-09-18}
7   {1.0}
8   {Enumerate exercise sheets}
```

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

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

12.4 Definition of variables

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

```

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

```

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

```

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

```

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

```

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

```

Internal variables used by functions `__enumext_is_not_nested:`, `__enumext_is_on_first_level:` and `__enumext_keyans_name_and_start:` (§12.5.1).

```

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

```

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

```

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

```

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

The initial values of these variables are set by the function `__enumext_define_counters:Nn` (§12.10) and then modified by the function `__enumext_label_style:Nnn` used by `label` key (§12.13).

```

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

```

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

```

\c__enumext_counter_style_tl
\l__enumext_ref_key_arg_tl
\l__enumext_ref_the_count_tl
\l__enumext_the_counter_X_tl
\l__enumext_renew_the_count_X_tl

```

Internal variables used by `ref` key (§12.13).

```

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

```

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

```

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

```

Internal variables used by `resume`, `resume*` and `series` keys (§12.24).

```

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

```

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

```

\l__enumext_current_widest_dim
\g__enumext_counter_styles_tl
\g__enumext_widest_label_tl
\l__enumext_label_width_by_box

```

The variable `\l__enumext_current_widest_dim` stores the current label width, the variable `\g__enumext_counter_styles_tl` stores the default *⟨label style⟩* and the variable `\g__enumext_widest_label_tl` the label width. These variables are used by `widest` (§12.14) and `label` (§12.12) keys.

```

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

```


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

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

The boolean variable `\l__enumext_leftmargin_tmp_X_bool` and the dimensional variable `\l__enumext_leftmargin_tmp_X_dim` are used by the `list-indent` key (§12.17). The variables `\l__enumext_leftmargin_X_dim` and `\l__enumext_itemindent_X_dim` are used and set by the function `__enumext_calc_hspace`:NNNNNNNNNN (§12.37.1).

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

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

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

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

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

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

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

Internal variables used by `\miniright` command (§12.22.4) and the keys `mini-right`, `mini-right*`, `mini-env` and `mini-sep` (§12.20, §12.22).

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

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

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

The bool vars `\l__enumext_wrap_label_X_bool` and `\l__enumext_wrap_label_opt_X_bool` are used by `wrap-label` and `wrap-label*` keys (§12.12), the integer `\l__enumext_start_X_int` are used by the `start` and `start*` keys (§12.14), the token list `\l__enumext_fake_item_indent_X_tl` is used by `itemindent` key (§12.17.1), the variables `\l__enumext_label_fill_left_X_tl` and `\l__enumext_label_fill_right_X_tl` are used by the `align` key (§12.12). The boolean vars `\l__enumext_vspace_a_star_X_bool`, `\l__enumext_vspace_b_star_X_bool` are used by `above`, `above*`, `below` and `below*` keys (§12.19).

```
98 \cs_set_protected:Npn \__enumext_tmp:n #1
99 {
100   \bool_new:c { \l__enumext_wrap_label_#1_bool }
101   \bool_new:c { \l__enumext_wrap_label_opt_#1_bool }
102   \int_new:c { \l__enumext_start_#1_int }
103   \tl_new:c { \l__enumext_fake_item_indent_#1_tl }
104   \tl_new:c { \l__enumext_label_fill_left_#1_tl }
105   \tl_new:c { \l__enumext_label_fill_right_#1_tl }
106   \bool_new:c { \l__enumext_vspace_a_star_#1_bool }
107   \bool_new:c { \l__enumext_vspace_b_star_#1_bool }
108 }
109 \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }
```

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

```

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

```

The variable `\l__enumext_store_active_bool` setting by `save-ans` key (§12.25.1) activates all the mechanism related to `\anskey`, `anskey*`, `keyans`, `keyans*` and `keyanspic` environments.

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

The variable `\l__enumext_store_anskey_arg_tl` save the *argument* of `\anskey` (§12.29) and the variables `\l__enumext_store_anskey_env_tl` and `\l__enumext_store_anskey_opt_tl` save the *⟨body⟩* and the *⟨keys⟩* of the environment `anskey*` (§12.30).

The variables `\l__enumext_store_current_label_tl` and `\l__enumext_store_current_opt_arg_tl` save the *current label* and *optional argument* of `\item*` (§12.36) and `\anspic*` (§12.41.2) for the `keyans`, `keyans*` and `keyanspic` environments.

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

```

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

```

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

```

\l__enumext_setkey_tmpa_tl
\l__enumext_setkey_tmpb_tl
\l__enumext_setkey_tmpa_int
\l__enumext_setkey_tmpa_seq
\l__enumext_setkey_tmpb_seq

```

Internal variables used by the command `\setenumext` (§12.47).

```

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

```

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

```

\l__enumext_meta_path_tl
\l__enumext_foreach_print_seq
\l__enumext_foreach_name_prop_tl
\g__enumext_foreach_default_keys_tl

```

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

```

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

```

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

```

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

```

Internal variables used by command `\printkeyans` (§12.46), `show-pos` key (§12.26), `item-sym*` key (§12.34), `save-key` key (§12.26.2) and “*storage level system*”.

```

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

```

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

```

\l__enumext_keyans_pic_body_seq
\l__enumext_keyans_pic_width_dim
\l__enumext_keyans_pic_above_int
\l__enumext_keyans_pic_below_int
\l__enumext_keyans_pic_above_skip
\l__enumext_keyans_pic_star_bool
\l__enumext_keyans_pic_label_pos_str
\l__enumext_anspic_label_box
\l__enumext_anspic_body_box
\l__enumext_anspic_label_htdp_dim
\l__enumext_anspic_body_htdp_dim

```

Internal variables used by `keyanspic` environment and `\anspic` command (§12.41.1).

```

139 \seq_new:N \l__enumext_keyans_pic_body_seq
140 \dim_new:N \l__enumext_keyans_pic_width_dim
141 \int_new:N \l__enumext_keyans_pic_above_int
142 \int_new:N \l__enumext_keyans_pic_below_int
143 \skip_new:N \l__enumext_keyans_pic_above_skip
144 \bool_new:N \l__enumext_keyans_pic_star_bool
145 \str_new:N \l__enumext_keyans_pic_label_pos_str
146 \box_new:N \l__enumext_anspic_label_box
147 \box_new:N \l__enumext_anspic_body_box
148 \dim_new:N \l__enumext_anspic_label_htdp_dim
149 \dim_new:N \l__enumext_anspic_body_htdp_dim

```

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

Internal variables used by “*internal check answer*” mechanism (§12.25.3) used by the `check-ans` and `no-store` keys and check for starred commands `\item*` in `keyans` and `keyans*` environments and `\anspic*` in `keyanspic` environment.

```

150 \bool_new:N \l__enumext_check_answers_bool
151 \bool_new:N \g__enumext_check_ans_key_bool
152 \tl_new:N \l__enumext_check_start_line_env_tl
153 \int_new:N \g__enumext_check_starred_cmd_int
154 \int_new:N \g__enumext_item_anskey_int
155 \int_new:N \g__enumext_item_number_int
156 \bool_new:N \l__enumext_item_number_bool
157 \int_new:N \g__enumext_item_answer_diff_int

```

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

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

```

158 \bool_new:N \l__enumext_hyperref_bool
159 \bool_new:N \l__enumext_footnotes_key_bool

```

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

Internal variables used by `save-ref` key (§12.26). The variables `\l__enumext_label_copy_X_tl` correspond to temporary copies of the $\langle labels \rangle$ defined by level on which operations will be performed.

The variables `\l__enumext_newlabel_arg_one_tl` and `\l__enumext_newlabel_arg_two_tl` will be used to form the arguments passed to the function `__enumext_newlabel:nn` (§12.8) and the variable `\l__enumext_write_aux_file_tl` will be in charge of executing the writing code in the `.aux` file.

```

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

```

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

Internal variables used for redefinition of `\footnote` (§12.42.1).

```

168 \int_new:N \g__enumext_footnote_int
169 \seq_new:N \g__enumext_footnote_arg_seq
170 \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`.)

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

```

171 \cs_set_protected:Npn \__enumext_tmp:n #1
172 {
173   \bool_new:c { l__enumext_item_starred_#1_bool }
174   \int_new:c { l__enumext_item_column_pos_#1_int }
175   \int_new:c { g__enumext_item_count_all_#1_int }
176   \int_new:c { l__enumext_joined_item_#1_int }
177   \int_new:c { l__enumext_joined_item_aux_#1_int }
178   \int_new:c { l__enumext_tmpa_#1_int }
179   \dim_new:c { l__enumext_tmpa_#1_dim }
180   \box_new:c { l__enumext_item_text_#1_box }
181   \dim_new:c { l__enumext_joined_width_#1_dim }
182   \dim_new:c { l__enumext_item_width_#1_dim }
183   \tl_new:c { g__enumext_item_symbol_aux_#1_tl }
184   \str_new:c { l__enumext_align_label_#1_str }
185   \bool_new:c { g__enumext_minipage_active_#1_bool }
186   \box_new:c { l__enumext_miniright_code_#1_box }
187   \bool_new:c { g__enumext_minipage_center_#1_bool }
188   \dim_new:c { g__enumext_minipage_right_#1_dim }
189   \skip_new:c { g__enumext_minipage_right_#1_skip }
190 }
191 \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
192 \clist_const:Nn \c__enumext_all_envs_clist
193 {
194     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
195     {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
196 }
```

(End of definition for `\c__enumext_all_envs_clist`.)

12.5 Some utility functions

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

```
197 \cs_generate_variant:Nn \keys_precompile:nnN { neN }
198 \cs_generate_variant:Nn \seq_use:Nn { NV }
```

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

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

```
199 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
200 {
201     \hook_gput_code:nnn {begindocument} {enumext} { #1 }
202 }
```

(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.

```
203 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
204 {
205     \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
206 }
207 \cs_new_protected:Npn \__enumext_before_env:nn #1 #2
208 {
209     \hook_gput_code:nnn {env/#1/before} {enumext} {#2}
210 }
```

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

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

```
211 \cs_new:Nn \__enumext_level:
212 {
213     \int_to_roman:n { \l__enumext_level_int }
214 }
```

(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?](#).

```
215 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
216 {
217     \regex_match:nnTF { ^[\+|-]?[\d]+$ } {#1} % $
218     { \prg_return_true: }
219     { \prg_return_false: }
220 }
```

(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.

```
221 \cs_new_protected:Nn \__enumext_regex_counter_style:
222 {
223     \tl_map_inline:Nn \c__enumext_counter_style_tl
224     {
225         \regex_replace_once:nnN { \c{##1}\* }
226         { \c{##1}\cB{\u{l__enumext_ref_the_count_tl}\cE} } \l__enumext_ref_key_arg_tl
227     }
228 }
```

(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.

```

229 \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
230 {
231     * ~ #2
232     \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
233     = ~ \use:c { #1_use:c } { \__enumext_#2_#3_#1 } \\
234 }

```

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

`__enumext_unskip_unkern:`

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

```

235 \cs_new_protected:Npn \__enumext_unskip_unkern:
236 {
237     \int_case:nnT { \lastnodetype }
238     {
239         { 11 }
240         {
241             \typeout{SKIIIIIIIIIIIIIIIP}
242             \typeout{\the\lastskip}
243             \unskip
244         }
245         { 12 }
246         {
247             \typeout{KERRRRRRRRRRRRRRRRRN}
248             \typeout{\the\lastkern}
249             \unkern
250         }
251     }
252 }

```

(End of definition for `__enumext_unskip_unkern:`.)

12.5.1 Utilities for environments and levels

`__enumext_is_not_nested:`

The function `__enumext_is_not_nested:` set the variables `\g__enumext_standar_bool` and `\g__enumext_starred_bool` to “*true*” only if the environments `enumext` and `enumext*` are nested in each other and save the environment name in `\l__enumext_envir_name_tl`.

`__enumext_is_on_first_level:`

```

253 \cs_new_protected:Nn \__enumext_is_not_nested:
254 {
255     \str_case:en { \@currenvir }
256     {
257         {enumext}
258         {
259             \tl_set:Nn \l__enumext_envir_name_tl { enumext }
260             \bool_lazy_and:nnT
261             { \bool_not_p:n { \g__enumext_standar_bool } }
262             { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
263             {
264                 \bool_gset_true:N \g__enumext_standar_bool
265             }
266         }
267         {enumext*}
268         {
269             \tl_set:Nn \l__enumext_envir_name_tl { enumext* }
270             \bool_lazy_and:nnT
271             { \bool_not_p:n { \g__enumext_starred_bool } }
272             { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
273             {
274                 \bool_gset_true:N \g__enumext_starred_bool
275             }
276         }
277     }
278 }

```

The function `__enumext_is_on_first_level:` will set the variables `\l__enumext_standar_first_bool` (§12.25.1), `\l__enumext_starred_first_bool` (§12.25.1) and `\l__enumext_anskey_env_bool` (§12.30) to “*true*” only if the environment is not nested and we are in the “*first level*” of it . We will also save the *start line number* of each environment in the variable `\g__enumext_start_line_tl` and the *name*

of each environment in the variable `\g__enumext_envir_name_tl` to use in messages related to the `check-ans` key and `.log` file.

```

279 \cs_new_protected:Nn \__enumext_is_on_first_level:
280 {
281   \bool_lazy_all:nT
282   {
283     { \bool_if_p:N \g__enumext_standar_bool }
284     { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
285     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
286   }
287   {
288     \bool_set_true:N \l__enumext_standar_first_bool
289     \bool_set_true:N \l__enumext_anskey_env_bool
290     \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
291     \tl_gset:Ne \g__enumext_start_line_tl
292       {
293         on ~ line ~ \exp_not:V \inputlineno
294       }
295   }
296   \bool_lazy_all:nT
297   {
298     { \bool_if_p:N \g__enumext_starred_bool }
299     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
300     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
301   }
302   {
303     \bool_set_true:N \l__enumext_starred_first_bool
304     \bool_set_true:N \l__enumext_anskey_env_bool
305     \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
306     \tl_gset:Ne \g__enumext_start_line_tl
307       {
308         on ~ line ~ \exp_not:V \inputlineno
309       }
310   }
311 }

```

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

`__enumext_keyans_name_and_start:`

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

```

312 \cs_new_protected:Nn \__enumext_keyans_name_and_start:
313 {
314   \str_case:en { \@currentenvir }
315   {
316     {keyans}
317     {
318       \tl_set:Nn \l__enumext_envir_name_tl { keyans }
319       \tl_set:Ne \l__enumext_check_start_line_env_tl
320         {
321           in ~ 'keyans' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
322         }
323     }
324     {keyans*}
325     {
326       \tl_set:Nn \l__enumext_envir_name_tl { keyans* }
327       \tl_set:Ne \l__enumext_check_start_line_env_tl
328         {
329           in ~ 'keyans*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
330         }
331     }
332     {keyanspic}
333     {
334       \tl_set:Nn \l__enumext_envir_name_tl { keyanspic }
335       \tl_set:Ne \l__enumext_check_start_line_env_tl
336         {
337           in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
338         }
339     }
340   }
341 }

```


(End of definition for `__enumext_keyans_name_and_start:`.)

12.5.2 Utilities for log and terminal

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

```
\__enumext_reset_global_vars:
\__enumext_reset_global_int:
\__enumext_reset_global_bool:
\__enumext_reset_global_tl:
342 \cs_new_protected:Nn \__enumext_reset_global_vars:
343 {
344   \__enumext_reset_global_int:
345   \__enumext_reset_global_bool:
346   \__enumext_reset_global_tl:
347 }
348 \cs_new_protected:Nn \__enumext_reset_global_int:
349 {
350   \int_gzero:N \g__enumext_item_number_int
351   \int_gzero:N \g__enumext_item_anskey_int
352   \int_gzero:N \g__enumext_item_answer_diff_int
353 }
354 \cs_new_protected:Nn \__enumext_reset_global_bool:
355 {
356   \bool_gset_false:N \g__enumext_check_ans_key_bool
357   \bool_gset_false:N \g__enumext_standar_bool
358   \bool_gset_false:N \g__enumext_starred_bool
359 }
360 \cs_new_protected:Nn \__enumext_reset_global_tl:
361 {
362   \tl_gclear:N \g__enumext_store_name_tl
363   \tl_gclear:N \g__enumext_start_line_tl
364   \tl_gclear:N \g__enumext_envir_name_tl
365 }
```

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

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

```
366 \cs_new_protected:Nn \__enumext_log_global_vars:
367 {
368   \msg_log:nneeee { enumext } { prop-seq-int-hook }
369   { \g__enumext_store_name_tl }
370   { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
371   { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
372   { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
373 }
```

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.

```
374 \cs_new_protected:Nn \__enumext_log_answer_vars:
375 {
376   \msg_log:nneee { enumext } { item-answer-hook }
377   { \int_use:N \g__enumext_item_number_int }
378   { \int_use:N \g__enumext_item_anskey_int }
379   { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int } }
380 }
```

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

12.6 Copying list and minipage environments

The `list` environment provided by L^AT_EX has the following plain form:

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

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

```
\__enumext_start_list:nn
\__enumext_stop_list:
\__enumext_item_std:w
381 \__enumext_at_begin_document:n
382 {
```

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.

```

383 \cs_new_eq:NN \__enumext_start_list:nn \list
384 \cs_new_eq:NN \__enumext_stop_list: \endlist
385 \NewCommandCopy \__enumext_item_std:w \item
386 }

```

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

The `minipage` environment provided by L^AT_EX has the following (simplified) plain form:

```

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

```

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

```

\__enumext_minipage:w
\__enumext_endminipage:

```

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

```

387 \__enumext_at_begin_document:n
388 {
389 \cs_new_eq:NN \__enumext_minipage:w \minipage
390 \cs_new_eq:NN \__enumext_endminipage: \endminipage
391 }

```

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

12.7 The internal minipage environment

```

\__enumext_internal_mini_page:
__enumext_mini_env*

```

The function `__enumext_internal_mini_page:` creates a internal `__enumext_mini_page` environment (custom version of `minipage`) setting the `\if@minipage` switch to “false” to allow spaces at the “above” of the environment, plus we will add `\skip_vertical:N \c_zero_skip` to maintain alignment on “top” in the first part and `\skip_vertical:N \c_zero_skip` in the second part to allow spaces “below”. This environment will be used internally by the `mini-env` key, it is not documented in the user interface and is for internal use only. This function is passed to the function `__enumext_safe_exec:` in the `enumext` environment definition (§12.38) and `__enumext_safe_exec_vii:` in the `enumext*` environment definition (§12.43)

```

392 \cs_new_protected:Nn \__enumext_internal_mini_page:
393 {
394 \int_compare:nNt { \l__enumext_level_int } = { 0 }
395 {
396 \DeclareDocumentEnvironment{__enumext_mini_page}{ m }
397 {
398 \__enumext_minipage:w [ t ] { ##1 }
399 \legacy_if_gset_false:n { @minipage }
400 \skip_vertical:N \c_zero_skip
401 }
402 {
403 \skip_vertical:N \c_zero_skip
404 \__enumext_endminipage:
405 }
406 }
407 }

```

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

12.8 Compatibility with hyperref and footnotehyper

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

```

408 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
409 \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”.

```

410 \cs_new_protected:Nn \__enumext_after_hyperref:
411 {
412 \IfPackageLoadedTF { hyperref }
413 {
414 \msg_info:nnn { enumext } { package-load } { hyperref }
415 \bool_set_true:N \l__enumext_hyperref_bool
416 \IfHyperBoolean{hyperfootnotes}
417 {

```

```

418         \typeout{hyperfootnotes=true}
419         \bool_set_true:N \l__enumext_footnotes_key_bool
420     }
421     { \typeout{hyperfootnotes=false} }
422 }
423 { }

```

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`.

```

424 \bool_if:NT \l__enumext_footnotes_key_bool
425 {
426     \IfPackageLoadedTF { footnotehyper }
427     {
428         \msg_info:nnn { enumext } { package-load } { footnotehyper }
429     }
430     {
431         \typeout{No ~ footnotehyper ~ load}
432         \typeout{Load ~ and ~ use ~ \string\makesavenoteenv{enumext*}}
433         \bool_set_false:N \l__enumext_footnotes_key_bool
434     }
435 }

```

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.

```

436 \bool_if:NTF \l__enumext_hyperref_bool
437 {
438     \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
439     \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
440 }
441 {
442     \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
443     \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
444 }
445 }

```

(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.

```

446 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
447 {
448     \protected@write \@auxout { }
449     {
450         \token_to_str:N \newlabel {#1}
451         {
452             {#2}
453             \bool_if:NT \l__enumext_hyperref_bool
454             { { \thepage } {#2} {#1} }
455             { }
456         }
457     }
458     \__enumext_hypertarget:nn {#1} { }
459     \__enumext_phantomsection:
460 }

```

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

12.9 Definition of public dimension

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

```

461 \dim_zero_new:N \itemwidth

```

12.10 Definition of counters

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

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

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

```
462 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
463 {
464   \cs_if_exist:cTF { c@ #2 }
465   { \msg_fatal:nnn { enumext } { counters }{ #2 } }
466   {
467     \tl_set:Nn #1 { #2 }
468     \newcounter { #2 }
469   }
470 }
```

(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 471 \__enumext_define_counters:Nn \__enumext_counter_i_tl { enumXi }
enumXii 472 \__enumext_define_counters:Nn \__enumext_counter_ii_tl { enumXii }
enumXiii 473 \__enumext_define_counters:Nn \__enumext_counter_iii_tl { enumXiii }
enumXiv 474 \__enumext_define_counters:Nn \__enumext_counter_iv_tl { enumXiv }
enumXv 475 \__enumext_define_counters:Nn \__enumext_counter_v_tl { enumXv }
enumXvi 476 \__enumext_define_counters:Nn \__enumext_counter_vi_tl { enumXvi }
enumXvii 477 \__enumext_define_counters:Nn \__enumext_counter_vii_tl { enumXvii }
enumXviii 478 \__enumext_define_counters:Nn \__enumext_counter_viii_tl { enumXviii }
```

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

12.11 Definition of labels

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

```
\__enumext_register_counter_style:Nn
```

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

```
479 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
480 {
481   \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
482   \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
483 }
484 \__enumext_register_counter_style:Nn \arabic { 0 }
485 \__enumext_register_counter_style:Nn \Alph { M }
486 \__enumext_register_counter_style:Nn \alph { m }
487 \__enumext_register_counter_style:Nn \Roman { VIII }
488 \__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.

```
489 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
490 {
491   \hbox_set:Nn \l__enumext_label_width_by_box {#2}
492   \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
493 }
494 \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.

```
495 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
```

```

496 {
497   \tl_clear_new:N #1
498   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
499   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
500   \tl_map_inline:Nn \g__enumext_counter_styles_tl
501   {
502     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
503     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
504     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
505   }
506   \__enumext_label_width_by_box:Nn \__enumext_current_widest_dim
507   { \tl_use:N \g__enumext_widest_label_tl }
508   \tl_set_eq:cN { the #2 } #1
509 }
510 \cs_generate_variant:Nn \__enumext_label_style:Nnn { cvn }

```

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

12.12 Setting keys associated with label

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

```

511 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
512 {
513   \keys_define:nn { enumext / #1 }
514   {
515     font .tl_set:c = { l__enumext_label_font_style_#2_tl },
516     font .value_required:n = true,
517     labelsep .dim_set:c = { l__enumext_labelsep_#2_dim },
518     labelsep .initial:n = {0.3333em},
519     labelsep .value_required:n = true,
520     labelwidth .dim_set:c = { l__enumext_labelwidth_#2_dim },
521     labelwidth .value_required:n = true,
522     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
523     wrap-label .initial:n = {##1},
524     wrap-label .value_required:n = true,
525     wrap-label* .code:n = {
526       \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
527       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
528     },
529     wrap-label* .value_required:n = true,
530   }
531 }
532 \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.

```

533 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
534 {
535   \keys_define:nn { enumext / #1 }
536   {
537     align .choice:,
538     align / left .code:n =
539     {
540       \tl_clear:c { l__enumext_label_fill_left_#2_tl }
541       \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
542     },
543     align / right .code:n =
544     {
545       \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
546       \tl_clear:c { l__enumext_label_fill_right_#2_tl }
547     },
548     align / center .code:n =
549     {
550       \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
551       \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
552     },

```

```

553         align / unknown .code:n =
554             \msg_error:nneee { enumext } { unknown-choice }
555             { align } { left, ~ right, ~ center } { \exp_not:n {##1} },
556         align .initial:n = left,
557         align .value_required:n = true,
558     }
559 }
560 \clist_map_inline:nn
561 {
562     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
563 }
564 { \__enumext_tmp:nn #1 }

565 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
566 {
567     \keys_define:nn { enumext / #1 }
568     {
569         align .choice:,
570         align / left .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
571         align / right .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
572         align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
573         align / unknown .code:n =
574             \msg_error:nneee { enumext } { unknown-choice }
575             { align } { left, ~ right, ~ center } { \exp_not:n {##1} },
576         align .initial:n = left,
577         align .value_required:n = true,
578     }
579 }
580 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

(End of definition for *align*.)

12.13 Setting label and ref keys

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

12.13.1 Define and set label and ref keys for enumext environment

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

```

\l__enumext_label_i_tl
\l__enumext_label_ii_tl
\l__enumext_label_iii_tl
\l__enumext_label_iv_tl

581 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
582 {
583     \keys_define:nn { enumext / #1 }
584     {
585         label .code:n = {
586             \__enumext_label_style:cnv { l__enumext_label_#2_tl }
587             { l__enumext_counter_#2_tl } {##1}
588             \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
589             \l__enumext_current_widest_dim
590         },
591         label .initial:n = #3,
592         label .value_required:n = true,
593         ref .code:n = \__enumext_standar_ref:n {##1},
594         ref .value_required:n = true,
595     }
596 }
597 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
598 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*) }
599 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
600 \__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }

```

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

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.

```

601 \cs_new_protected:Npn \__enumext_standar_ref:n #1

```



```

602 {
603   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
604   \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
605   {
606     \msg_error:nnn { enumext } { key-ref-empty } { enumext }
607   }
608   {
609     \tl_set_eq:Nc
610     \l__enumext_ref_the_count_tl { \l__enumext_counter_ \__enumext_level: _tl }
611     \__enumext_regex_counter_style:
612     \tl_set_eq:Nc
613     \l__enumext_ref_the_count_tl { \l__enumext_the_counter_ \__enumext_level: _tl }
614     \tl_put_right:ce { \l__enumext_renew_the_count_ \__enumext_level: _tl }
615     {
616       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
617       { \exp_not:V \l__enumext_ref_key_arg_tl }
618     }
619   }
620 }

```

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

```

621 \cs_new_protected:Nn \__enumext_standar_ref:
622 {
623   \tl_if_empty:cF { \l__enumext_renew_the_count_ \__enumext_level: _tl }
624   {
625     \tl_use:c { \l__enumext_renew_the_count_ \__enumext_level: _tl }
626   }
627 }

```

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

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

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

```

\l__enumext_label_vii_tl
\l__enumext_label_viii_tl
628 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
629 {
630   \keys_define:nn { enumext / #1 }
631   {
632     label .code:n = {
633       \__enumext_label_style:cvn { \l__enumext_label_#2_tl }
634       { \l__enumext_counter_#2_tl } {##1}
635       \dim_set_eq:cN { \l__enumext_labelwidth_#2_dim }
636       \l__enumext_current_widest_dim
637     },
638     label .initial:n = #3,
639     label .value_required:n = true,
640     ref .code:n = \__enumext_starred_ref:n {##1},
641     ref .value_required:n = true,
642   }
643 }
644 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
645 \__enumext_tmp:nnn { keyans* } { viii } { \Alph*.}

```

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

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

```

646 \cs_new_protected:Npn \__enumext_starred_ref:n #1
647 {
648   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
649   \int_compare:nNt { \l__enumext_level_h_int } = { 1 }
650   {
651     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
652     {
653       \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
654     }
655     {
656       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
657       \__enumext_regex_counter_style:
658       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
659       \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl

```

```

660         {
661             \exp_not:N \renewcommand { \exp_not:V \__enumext_ref_the_count_tl }
662             { \exp_not:V \__enumext_ref_key_arg_tl }
663         }
664     }
665 }
666 \int_compare:nNnT { \__enumext_keyans_level_h_int } = { 1 }
667 {
668     \tl_if_empty:NTF \__enumext_ref_key_arg_tl
669     {
670         \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
671     }
672     {
673         \tl_set_eq:NN \__enumext_ref_the_count_tl \__enumext_counter_viii_tl
674         \__enumext_regex_counter_style:
675         \tl_set_eq:NN \__enumext_ref_the_count_tl \__enumext_the_counter_viii_tl
676         \tl_put_right:Ne \__enumext_renew_the_count_viii_tl
677         {
678             \exp_not:N \renewcommand { \exp_not:V \__enumext_ref_the_count_tl }
679             { \exp_not:V \__enumext_ref_key_arg_tl }
680         }
681     }
682 }
683 }

```

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.

```

684 \cs_new_protected:Nn \__enumext_starred_ref:
685 {
686     \int_compare:nNnT { \__enumext_level_h_int } = { 1 }
687     {
688         \tl_if_empty:NF \__enumext_renew_the_count_vii_tl
689         {
690             \tl_use:N \__enumext_renew_the_count_vii_tl
691         }
692     }
693     \int_compare:nNnT { \__enumext_keyans_level_h_int } = { 1 }
694     {
695         \tl_if_empty:NF \__enumext_renew_the_count_viii_tl
696         {
697             \tl_use:N \__enumext_renew_the_count_viii_tl
698         }
699     }
700 }

```

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

12.13.3 Define and set label and ref keys for keyans and keyanspic environments

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

```

\__enumext_label_v_tl
\__enumext_label_vi_tl
701 \keys_define:nn { enumext / keyans }
702 {
703     label .code:n = {
704         \__enumext_label_style:cvn { \__enumext_label_v_tl }
705         { \__enumext_counter_v_tl } {#1}
706         \dim_set_eq:cN { \__enumext_labelwidth_v_dim }
707         \__enumext_current_widest_dim
708         \__enumext_label_style:cvn { \__enumext_label_vi_tl }
709         { \__enumext_counter_vi_tl } {#1}
710         \dim_set_eq:cN { \__enumext_labelwidth_v_dim }
711         \__enumext_current_widest_dim
712     },
713     label .initial:n = \Alph*,
714     label .value_required:n = true,
715     ref .code:n = \__enumext_keyans_ref:n {#1},
716     ref .value_required:n = true,
717 }

```

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

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

```

718 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
719 {
720   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
721   \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
722   {
723     \msg_error:nnn { enumext } { key-ref-empty } { keyans }
724   }
725   {
726     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
727     \__enumext_regex_counter_style:
728     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
729     \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
730     {
731       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
732       { \exp_not:V \l__enumext_ref_key_arg_tl }
733     }
734   }
735 }

```

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

```

736 \cs_new_protected:Nn \__enumext_keyans_ref:
737 {
738   \tl_if_empty:NF \l__enumext_renew_the_count_v_tl
739   {
740     \tl_use:N \l__enumext_renew_the_count_v_tl
741   }
742 }

```

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

12.14 Setting start, start* and widest keys

```

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

```

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

```

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

```

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

```

743 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
744 {
745   \__enumext_if_is_int:nTF { #3 }
746   {
747     \int_set:Nn #2 {#3}
748   }
749   {
750     \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
751     { \int_set:Nn #2 { \int_from_alph:n {#3} } }
752     \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
753     { \int_set:Nn #2 { \int_from_roman:n {#3} } }
754   }
755 }
756 \cs_generate_variant:Nn \__enumext_start_from:NNn { ccn, cce }

```

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

```

\__enumext_widest_from:nNNn
\__enumext_widest_from:nccn

```

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

```

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

```

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

```

757 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
758 {
759   \__enumext_if_is_int:nTF {#4}
760   {
761     \setcounter{enumX#1} { #4 }

```

```

762     }
763     {
764         \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
765         { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
766         \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
767         { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
768     }
769     \__enumext_label_width_by_box:cv
770     { l__enumext_labelwidth_#1_dim } { l__enumext_label_#1_tl }
771 }
772 \cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }

```

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

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

```

773 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
774 {
775     \keys_define:nn { enumext / #1 }
776     {
777         start* .code:n = {
778             \__enumext_start_from:ccn
779             { l__enumext_label_#2_tl }
780             { l__enumext_start_#2_int } {##1}
781         },
782         start* .value_required:n = true,
783         start .code:n = {
784             \__enumext_start_from:cce
785             { l__enumext_label_#2_tl }
786             { l__enumext_start_#2_int } { \int_eval:n {##1} }
787         },
788         start .initial:n = 1,
789         start .value_required:n = true,
790         widest .code:n = {
791             \__enumext_widest_from:nccn {#2}
792             { l__enumext_label_#2_tl }
793             { l__enumext_labelwidth_#2_dim } {##1}
794         },
795         widest .value_required:n = true,
796     }
797 }
798 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

12.15 Setting keys for vertical spaces

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

```

799 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
800 {
801     \keys_define:nn { enumext / #1 }
802     {
803         topsep .skip_set:c = { l__enumext_topsep_#2_skip },
804         topsep .initial:n = {#3},
805         topsep .value_required:n = true,
806         partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
807         partopsep .initial:n = {#4},
808         partopsep .value_required:n = true,
809         parsep .skip_set:c = { l__enumext_parsep_#2_skip },
810         parsep .initial:n = {#5},
811         parsep .value_required:n = true,
812         itemsep .skip_set:c = { l__enumext_itemsep_#2_skip },
813         itemsep .initial:n = {#6},
814         itemsep .value_required:n = true,
815         noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
816         noitemsep .value_forbidden:n = true,
817         nosep .meta:n = {
818             itemsep = 0pt, parsep = 0pt,
819             topsep = 0pt, partopsep = 0pt,
820         },

```

```

821         nosep      .value_forbidden:n = true,
822     }
823 }

```

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

```

824 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
825 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
826 { 4.0pt plus 2.0pt minus 1.0pt }
827 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
828 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
829 { 2.0pt plus 1.0pt minus 1.0pt }
830 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
831 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
832 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
833 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
834 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
835 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
836 { 2.0pt plus 1.0pt minus 1.0pt }
837 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
838 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
839 { 4.0pt plus 2.0pt minus 1.0pt }
840 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
841 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
842 { 2.0pt plus 1.0pt minus 1.0pt }

```

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

12.16 Setting base-fix key

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

```

base-fix \__enumext_nested_base_line_fix:
843 \cs_set_protected:Npn \__enumext_tmp:n #1
844 {
845     \keys_define:nn { enumext / #1 }
846     {
847         base-fix .bool_set:N = \l__enumext_base_line_fix_bool,
848         base-fix .initial:n = false,
849         base-fix .value_forbidden:n = true,
850     }
851 }
852 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

```

853 \cs_new_protected:Nn \__enumext_nested_base_line_fix:
854 {
855     \bool_lazy_and:nnT
856     { \bool_if_p:N \l__enumext_standar_first_bool }
857     { \bool_if_p:N \l__enumext_base_line_fix_bool }
858     {
859         \mode_leave_vertical:
860         \vspace { -\baselineskip }
861         \keys_set:nn { enumext / level-1 }
862         {
863             topsep = 0pt, above = 0pt, above* = 0pt,
864         }
865     }
866     \bool_lazy_and:nnT
867     { \bool_if_p:N \l__enumext_starred_first_bool }
868     { \bool_if_p:N \l__enumext_base_line_fix_bool }
869     {
870         \mode_leave_vertical:
871         \vspace { -\baselineskip }
872         \keys_set:nn { enumext / enumext* }
873         {
874             topsep = 0pt, above = 0pt, above* = 0pt,

```

```

875     }
876   }
877   \bool_set_false:N \__enumext_base_line_fix_bool
878 }

```

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

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

12.17 Setting keys for horizontal spaces

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

```

879 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
880 {
881   \keys_define:nn { enumext / #1 }
882   {
883     itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
884     itemindent .value_required:n = true,
885     rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
886     rightmargin .value_required:n = true,
887     listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
888     listparindent .value_required:n = true,
889     list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
890     list-offset .value_required:n = true,
891     list-indent .code:n =
892       \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
893       \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
894     list-indent .value_required:n = true,
895   }
896 }
897 \clist_map_inline:nn
898 {
899   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
900 }
901 { \__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.

```

902 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
903 {
904   \keys_define:nn { enumext / #1 }
905   {
906     itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
907     itemindent .value_required:n = true,
908     rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
909     rightmargin .value_required:n = true,
910     listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
911     listparindent .value_required:n = true,
912     list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
913     list-offset .value_required:n = true,
914     list-indent .meta:n = { list-offset = ##1 },
915     list-indent .value_required:n = true,
916   }
917 }
918 \clist_map_inline:nn
919 {
920   {enumext*}{vii}, {keyans*}{viii}
921 }
922 { \__enumext_tmp:nn #1 }

```

12.17.1 Functions for setting the fake `itemindent`

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

```

923 \cs_set_protected:Nn \__enumext_fake_item:
924 {
925   \dim_compare:nNt
926   { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
927   >

```



```

928     { \c_zero_dim }
929     {
930         \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
931         {
932             \exp_not:N \mode_leave_vertical:
933             \exp_not:n { \skip_horizontal:n }
934             { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
935             \ignorespaces
936         }
937     }
938 }
939 \cs_set_protected:Nn \__enumext_keyans_fake_item:
940 {
941     \dim_compare:nNnT
942     { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
943     {
944         \tl_set:Ne \l__enumext_fake_item_indent_v_tl
945         {
946             \exp_not:N \mode_leave_vertical:
947             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
948         }
949     }
950 }
951 \cs_set_protected:Nn \__enumext_fake_item_vii:
952 {
953     \dim_compare:nNnT
954     { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
955     {
956         \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
957         {
958             \exp_not:N \mode_leave_vertical:
959             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
960         }
961     }
962 }
963 \cs_set_protected:Nn \__enumext_fake_item_viii:
964 {
965     \dim_compare:nNnT
966     { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
967     {
968         \tl_set:Ne \l__enumext_fake_item_indent_viii_tl
969         {
970             \exp_not:N \mode_leave_vertical:
971             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
972         }
973     }
974 }

```

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

12.18 Setting show-length key

show-length

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

```

975 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
976 {
977     \keys_define:nn { enumext / #1 }
978     {
979         show-length .bool_set:c = { l__enumext_show_length_#2_bool },
980         show-length .initial:n = false,
981     }
982 }
983 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

12.19 Setting before, after and first keys

before
before*
after
first

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

```

984 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
985 {
986   \keys_define:nn { enumext / #1 }
987   {
988     before .tl_set:c = { l__enumext_before_no_starred_key_#2_tl },
989     before .value_required:n = true,
990     before* .tl_set:c = { l__enumext_before_starred_key_#2_tl },
991     before* .value_required:n = true,
992     after .tl_set:c = { l__enumext_after_stop_list_#2_tl },
993     after .value_required:n = true,
994     first .tl_set:c = { l__enumext_after_list_args_#2_tl },
995     first .value_required:n = true,
996   }
997 }
998 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

12.19.1 Functions for before, after and first keys in enumext

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

```

999 \cs_new_protected:Nn \__enumext_before_args_exec:
1000 {
1001   \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
1002 }

```

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

```

1003 \cs_new_protected:Nn \__enumext_before_keys_exec:
1004 {
1005   \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
1006 }

```

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

```

1007 \cs_new_protected:Nn \__enumext_after_stop_list:
1008 {
1009   \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
1010 }

```

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

```

1011 \cs_new_protected:Nn \__enumext_after_args_exec:
1012 {
1013   \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
1014 }

```

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

12.19.2 Functions for before, after and first keys in keyans

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

```

\__enumext_before_args_exec_v:
\__enumext_before_keys_exec_v:
\__enumext_after_stop_list_v:
\__enumext_after_args_exec_v:
1015 \cs_new_protected:Nn \__enumext_before_args_exec_v:
1016 {
1017   \tl_use:N \l__enumext_before_starred_key_v_tl
1018 }
1019 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
1020 {
1021   \tl_use:N \l__enumext_before_no_starred_key_v_tl
1022 }
1023 \cs_new_protected:Nn \__enumext_after_stop_list_v:
1024 {
1025   \tl_use:N \l__enumext_after_stop_list_v_tl
1026 }
1027 \cs_new_protected:Nn \__enumext_after_args_exec_v:
1028 {
1029   \tl_use:N \l__enumext_after_list_args_v_tl
1030 }

```

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

12.19.3 Functions for before, after and first keys in enumext* and keyans*

```
\__enumext_before_args_exec_vii:
\__enumext_before_keys_exec_vii
\__enumext_after_stop_list_vii:
\__enumext_after_args_exec_vii:
```

Same implementation as the one used in the [enumext](#) environment.

```
1031 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
1032 {
1033   \tl_use:N \l__enumext_before_starred_key_vii_tl
1034 }
1035 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
1036 {
1037   \tl_use:N \l__enumext_before_starred_key_viii_tl
1038 }
1039 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
1040 {
1041   \tl_use:N \l__enumext_before_no_starred_key_vii_tl
1042 }
1043 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
1044 {
1045   \tl_use:N \l__enumext_before_no_starred_key_viii_tl
1046 }
1047 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
1048 {
1049   \tl_use:N \l__enumext_after_stop_list_vii_tl
1050 }
1051 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
1052 {
1053   \tl_use:N \l__enumext_after_stop_list_viii_tl
1054 }
1055 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
1056 {
1057   \tl_use:N \l__enumext_after_list_args_vii_tl
1058 }
1059 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
1060 {
1061   \tl_use:N \l__enumext_after_list_args_viii_tl
1062 }
```

(End of definition for __enumext_before_args_exec_vii: and others.)

12.20 Setting keys for multicol 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.

```
1063 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1064 {
1065   \keys_define:nn { enumext / #1 }
1066   {
1067     mini-env .dim_set:c = { l__enumext_minipage_right_#2_dim },
1068     mini-env .value_required:n = true,
1069     mini-sep .dim_set:c = { l__enumext_minipage_hsep_#2_dim },
1070     mini-sep .initial:n = 0.3333em,
1071     mini-sep .value_required:n = true,
1072     columns-sep .dim_set:c = { l__enumext_columns_sep_#2_dim },
1073     columns-sep .value_required:n = true,
1074     columns .int_set:c = { l__enumext_columns_#2_int },
1075     columns .initial:n = 1,
1076     columns .value_required:n = true,
1077   }
1078 }
1079 \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.

```
1080 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1081 {
1082   \keys_define:nn { enumext / #1 }
1083   {
1084     mini-right .tl_gset:c = { g__enumext_miniright_code_#2_tl },
1085     mini-right .value_required:n = true,
1086     mini-right* .code:n = {
```

```

1087         \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
1088         \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1089     },
1090     mini-right* .value_required:n = true,
1091 }
1092 }
1093 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

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

12.21 Adjustment of vertical spaces for multicol

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



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

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

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

12.21.1 Adjustment of vertical spaces for multicol in enumext

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

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

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

```

1094 \cs_new_protected:Nn \__enumext_multi_set_vskip:
1095 {
1096     \skip_set:cn { \l__enumext_multicol_above_ \__enumext_level: _skip }
1097     {
1098         \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1099     }
1100     \skip_set:cn { \l__enumext_multicol_below_ \__enumext_level: _skip }
1101     {
1102         \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1103     }
1104     \__enumext_add_pre_parsep:
1105 }

```

(End of definition for `__enumext_multi_set_vskip:`)

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

```

1106 \cs_new_protected:Nn \__enumext_add_pre_parsep:
1107 {
1108     \int_case:nn { \l__enumext_level_int }
1109     {
1110         { 2 }{
1111             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1112             {
1113                 \skip_add:Nn \l__enumext_multicol_above_ii_skip { \l__enumext_parsep_i_skip }
1114             }
1115         }
1116         { 3 }{
1117             \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1118             {

```

```

1119         \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_ii_skip
1120     }
1121 }
1122 { 4 }{
1123     \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1124     {
1125         \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip
1126     }
1127     }
1128 }
1129 }

```

(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*.

```

1130 \cs_new_protected:Nn \__enumext_multi_addvspace:
1131 {
1132     \__enumext_multi_set_vskip:
1133     \mode_if_vertical:T
1134     {
1135         \skip_add:cn { \l__enumext_multicols_above_ \__enumext_level: _skip }
1136         {
1137             \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1138         }
1139         \skip_add:cn { \l__enumext_multicols_below_ \__enumext_level: _skip }
1140         {
1141             \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1142         }
1143     }
1144     %%\__enumext_unskip_unkern:
1145     \par\nopagebreak
1146     \addvspace{ \skip_use:c { \l__enumext_multicols_above_ \__enumext_level: _skip } }
1147 }

```

(End of definition for `__enumext_multi_addvspace:`)

12.21.2 Adjustment of vertical spaces for multicols in keyans

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

`__enumext_keyans_multi_addvspace:`

```

1148 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1149 {
1150     \skip_set:Nn \l__enumext_multicols_above_v_skip
1151     {
1152         \l__enumext_topsep_v_skip
1153     }
1154     \skip_set:Nn \l__enumext_multicols_below_v_skip
1155     {
1156         \l__enumext_topsep_v_skip
1157     }
1158 }
1159 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1160 {
1161     \__enumext_keyans_multi_set_vskip:
1162     \mode_if_vertical:T
1163     {
1164         \skip_add:Nn \l__enumext_multicols_above_v_skip
1165         {
1166             \skip_use:N \l__enumext_partopsep_v_skip
1167         }
1168         \skip_add:Nn \l__enumext_multicols_below_v_skip
1169         {
1170             \skip_use:N \l__enumext_partopsep_v_skip
1171         }
1172     }
1173     \__enumext_unskip_unkern:
1174     \par\nopagebreak
1175     \addvspace{ \l__enumext_multicols_above_v_skip }
1176 }

```

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

12.22 Adjustment of vertical spaces for minipage

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

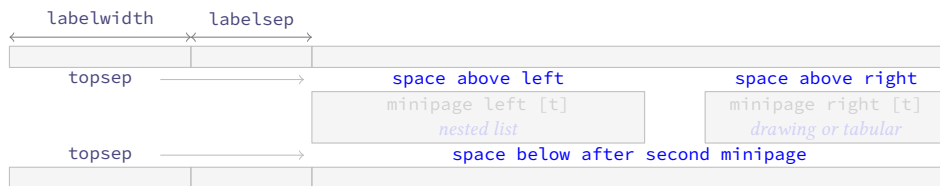


Figure 8: Representation of the `minipage` spacing adjustment for a nested level.

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

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

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

12.22.1 Adjustment of vertical spaces for minipage in enumext

`__enumext_minipage_set_skip:`
`__enumext_minipage_add_space:`

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

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

```

1177 \cs_new_protected:Nn \__enumext_minipage_set_skip:
1178 {
1179   \skip_set:Nn \__enumext_minipage_right_skip
1180   {
1181     \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1182   }
1183   \mode_if_vertical:T
1184   {
1185     \skip_add:Nn \__enumext_minipage_right_skip
1186     {
1187       \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1188     }
1189   }
1190   \skip_set_eq:NN \__enumext_minipage_after_skip \__enumext_minipage_right_skip

```

We will adjust the values `__enumext_multicols_above_X_skip` and `__enumext_multicols_below_X_skip` and call the function `__enumext_pre_itemsep_skip:`.

```

1191   \skip_set_eq:cN
1192   { l__enumext_multicols_above_ \__enumext_level: _skip } \__enumext_minipage_right_skip
1193   \skip_set_eq:cN
1194   { l__enumext_multicols_below_ \__enumext_level: _skip } \__enumext_minipage_right_skip
1195   \__enumext_pre_itemsep_skip:

```

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

```

1196   \int_compare:nNnT
1197   { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1198   {
1199     \skip_zero:N \topskip
1200     \skip_set_eq:Nc \multicolsep { l__enumext_multicols_above_ \__enumext_level: _skip }
1201   }
1202 }

```

The function `__enumext_minipage_add_space:` will apply the spaces on the “left side” using `\addvspace` “above” the `__enumext_mini_page` environment, taking into account whether \TeX is in *horizontal mode* or *vertical mode*. Here we use the plain \TeX macro `\nointerlineskip` to prevent baseline “glue” being

added between the next pair of boxes in a *vertical list*. For the latter we will make some adjustments since the `\partopsep` parameter comes into play and this affects the *vertical spacing*.

```

1203 \cs_new_protected:Nn \__enumext_minipage_add_space:
1204 {
1205   \__enumext_minipage_set_skip:
1206   \__enumext_unskip_unkern:
1207   \mode_if_vertical:TF
1208   {
1209     \nopagebreak\nointerlineskip
1210   }
1211   {
1212     \par\nopagebreak\nointerlineskip
1213     \skip_zero:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1214   }
1215   \int_compare:nNnTF
1216   { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1217   {
1218     \addvspace{ 0.445\box_ht:N \strutbox }
1219   }
1220   {
1221     \addvspace{ 0.250\box_ht:N \strutbox }
1222   }
1223 }

```

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

`__enumext_pre_itemsep_skip:`

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

```

1224 \cs_new_protected:Nn \__enumext_pre_itemsep_skip:
1225 {
1226   \int_case:nn { \l__enumext_level_int }
1227   {
1228     { 2 }{
1229       \skip_if_eq:nnTF
1230       { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1231       {
1232         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1233         \skip_set:Nn \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1234       }
1235       {
1236         \dim_compare:nNnT
1237         { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1238         {
1239           \skip_sub:Nn
1240           \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1241           \skip_sub:Nn
1242           \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1243           \skip_add:Nn
1244           \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1245           \skip_add:Nn
1246           \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1247         }
1248         \dim_compare:nNnT
1249         { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1250         {
1251           \skip_set:Nn \l_tmpa_skip
1252           {
1253             \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1254           }
1255           \skip_sub:Nn
1256           \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1257           \skip_sub:Nn
1258           \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1259           \skip_add:Nn
1260           \l__enumext_minipage_after_skip
1261           { 0.150\box_ht:N \strutbox + \l_tmpa_skip }
1262           \skip_add:Nn
1263           \l__enumext_multicols_below_ii_skip
1264           { 0.350\box_ht:N \strutbox + \l_tmpa_skip }
1265         }
1266       }
1267     }
1268   }

```

```

1266         }
1267     }
1268     { 3 }{
1269         \skip_if_eq:nnTF
1270         { \l__enumext_itemsep_ii_skip } { \c_zero_skip }
1271         {
1272             \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1273             \skip_set:Nn \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1274         }
1275         {
1276             \dim_compare:nnNt
1277             { \l__enumext_itemsep_ii_skip } < { \l__enumext_minipage_after_skip }
1278             {
1279                 \skip_sub:Nn
1280                 \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1281                 \skip_sub:Nn
1282                 \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1283                 \skip_add:Nn
1284                 \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1285                 \skip_add:Nn
1286                 \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1287             }
1288             \dim_compare:nnNt
1289             { \l__enumext_itemsep_ii_skip } > { \l__enumext_minipage_after_skip }
1290             {
1291                 \skip_set:Nn \l_tmpa_skip
1292                 {
1293                     \l__enumext_itemsep_ii_skip - \l__enumext_minipage_after_skip
1294                 }
1295                 \skip_sub:Nn
1296                 \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1297                 \skip_sub:Nn
1298                 \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1299                 \skip_add:Nn
1300                 \l__enumext_minipage_after_skip
1301                 { 0.150\box_ht:N \strutbox + \l_tmpa_skip }
1302                 \skip_add:Nn
1303                 \l__enumext_multicols_below_iii_skip
1304                 { 0.350\box_ht:N \strutbox + \l_tmpa_skip }
1305             }
1306         }
1307     }
1308     { 4 }{
1309         \skip_if_eq:nnTF { \l__enumext_itemsep_iii_skip } { \c_zero_skip }
1310         {
1311             \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1312             \skip_set:Nn \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1313         }
1314         {
1315             \dim_compare:nnNt
1316             { \l__enumext_itemsep_iii_skip } < { \l__enumext_minipage_after_skip }
1317             {
1318                 \skip_sub:Nn
1319                 \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1320                 \skip_sub:Nn
1321                 \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1322                 \skip_add:Nn
1323                 \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1324                 \skip_add:Nn
1325                 \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1326             }
1327             \dim_compare:nnNt
1328             { \l__enumext_itemsep_iii_skip } > { \l__enumext_minipage_after_skip }
1329             {
1330                 \skip_set:Nn \l_tmpa_skip
1331                 {
1332                     \l__enumext_itemsep_iii_skip - \l__enumext_minipage_after_skip
1333                 }
1334                 \skip_sub:Nn
1335                 \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1336                 \skip_sub:Nn

```

```

1337         \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1338     \skip_add:Nn
1339     \l__enumext_minipage_after_skip
1340     { 0.150\box_ht:N \strutbox + \l_tmpa_skip }
1341     \skip_add:Nn
1342     \l__enumext_multicols_below_iv_skip
1343     { 0.350\box_ht:N \strutbox + \l_tmpa_skip }
1344 }
1345 }
1346 }
1347 }
1348 }

```

(End of definition for `__enumext_pre_itemsep_skip:`)

12.22.2 Adjustment of vertical spaces for minipage in keyans

```

\__enumext_keyans_minipage_set_skip:
\__enumext_keyans_minipage_add_space:
\__enumext_keyans_pre_itemsep_skip:

```

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

```

1349 \cs_new_protected:Nn \__enumext_keyans_minipage_set_skip:
1350 {
1351     \skip_zero:N \l__enumext_minipage_after_skip
1352     \skip_zero:N \l__enumext_minipage_left_skip
1353     \skip_zero:N \l__enumext_minipage_right_skip
1354     \skip_set:Nn \l__enumext_minipage_right_skip
1355     {
1356         \l__enumext_topsep_v_skip
1357     }
1358     \mode_if_vertical:T
1359     {
1360         \skip_add:Nn \l__enumext_minipage_right_skip
1361         {
1362             \l__enumext_partopsep_v_skip
1363         }
1364     }
1365     \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip
1366     \skip_set_eq:NN \l__enumext_multicols_above_v_skip \l__enumext_minipage_right_skip
1367     \skip_set_eq:NN \l__enumext_multicols_below_v_skip \l__enumext_minipage_right_skip
1368     \__enumext_keyans_pre_itemsep_skip:
1369     \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
1370     {
1371         \skip_zero:N \topskip
1372         \skip_set_eq:NN \multicolsep \l__enumext_minipage_right_skip
1373     }
1374 }
1375 \cs_new_protected:Nn \__enumext_keyans_minipage_add_space:
1376 {
1377     \__enumext_keyans_minipage_set_skip:
1378     \__enumext_unskip_unkern:
1379     \mode_if_vertical:TF
1380     {
1381         \nopagebreak\nointerlineskip
1382     }
1383     {
1384         \par\nopagebreak\nointerlineskip
1385         \skip_zero:N \l__enumext_partopsep_v_skip
1386     }
1387     \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1388     {
1389         \addvspace{ 0.445\box_ht:N \strutbox }
1390     }
1391     {
1392         \addvspace{ 0.250\box_ht:N \strutbox }
1393     }
1394 }
1395 \cs_new_protected:Nn \__enumext_keyans_pre_itemsep_skip:
1396 {
1397     \skip_if_eq:nnTF
1398     { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1399     {
1400         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }

```

```

1401     \skip_set:Nn \l__enumext_multicols_below_v_skip { 0.35\box_ht:N \strutbox }
1402   }
1403   {
1404     \dim_compare:nNnT
1405       { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1406     {
1407       \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1408       \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1409       \skip_add:Nn \l__enumext_minipage_after_skip { 0.15\box_ht:N \strutbox }
1410       \skip_add:Nn \l__enumext_multicols_below_v_skip { 0.35\box_ht:N \strutbox }
1411     }
1412     \dim_compare:nNnT
1413       { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1414     {
1415       \skip_set:Nn \l_tmpa_skip
1416       {
1417         \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1418       }
1419       \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1420       \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1421       \skip_add:Nn \l__enumext_minipage_after_skip
1422         { 0.15\box_ht:N \strutbox + \l_tmpa_skip }
1423       \skip_add:Nn \l__enumext_multicols_below_v_skip
1424         { 0.35\box_ht:N \strutbox + \l_tmpa_skip }
1425     }
1426   }
1427 }

```

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

12.22.3 Adjustment of vertical spaces for minipage in enumext* and keyans*

`__enumext_mini_set_vskip_vii:`
`__enumext_mini_set_vskip_viii:`

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

```

1428 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1429 {
1430   \skip_zero_new:N \l__enumext_minipage_left_skip
1431   \skip_gzero_new:N \g__enumext_minipage_right_skip
1432   \skip_gzero_new:N \g__enumext_minipage_after_skip
1433   \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1434   {
1435     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1436     \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1437   }
1438   {
1439     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1440     \skip_gset:Nn \g__enumext_minipage_right_skip
1441       {
1442         \l__enumext_topsep_vii_skip
1443       }
1444     \skip_gset:Nn \g__enumext_minipage_after_skip
1445       {
1446         0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1447       }
1448   }
1449 }
1450 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1451 {
1452   \skip_zero_new:N \l__enumext_minipage_after_skip
1453   \skip_zero_new:N \l__enumext_minipage_left_skip
1454   \skip_zero_new:N \l__enumext_minipage_right_skip
1455   \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1456   {
1457     \skip_set:Nn \l__enumext_minipage_left_skip
1458       {
1459         0.5\box_dp:N \strutbox
1460       }
1461     \skip_set:Nn \l__enumext_minipage_right_skip
1462       {
1463         \l__enumext_partopsep_viii_skip

```

```

1464     }
1465     \skip_set:Nn \l__enumext_minipage_after_skip
1466     {
1467         1.6\box_dp:N \strutbox
1468     }
1469 }
1470 {
1471     \skip_set:Nn \l__enumext_minipage_left_skip
1472     {
1473         0.5875\box_dp:N \strutbox
1474     }
1475     \skip_set:Nn \l__enumext_minipage_right_skip
1476     {
1477         \l__enumext_topsep_viii_skip
1478     }
1479     \skip_set:Nn \l__enumext_minipage_after_skip
1480     {
1481         0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1482     }
1483 }
1484 }

```

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

`__enumext_mini_addvspace_vii:`
`__enumext_mini_addvspace_viii:`

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

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

```

1485 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1486 {
1487     \__enumext_mini_set_vskip_vii:
1488     \par\nopagebreak
1489     \addvspace { \l__enumext_minipage_left_skip }
1490 }
1491 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1492 {
1493     \__enumext_mini_set_vskip_viii:
1494     \par\nopagebreak
1495     \addvspace { \l__enumext_minipage_left_skip }
1496 }

```

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

12.22.4 The command `\miniright`

The command `\miniright` will close the `__enumext_mini_page` environment on the “*left side*”, open the `__enumext_mini_page` environment on the “*right side*” adding the *adjusted vertical space*. By default we will add `\centering` when starting the “*right side*” environment. The *starred argument* ‘`*`’ inhibits the use of `\centering` command i.e. the usual \TeX justification is maintained in the `__enumext_mini_page` on the “*right side*”.

`\miniright`

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

```

1497 \NewDocumentCommand \miniright { s }
1498 {
1499     \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
1500     {
1501         \msg_error:nnn { enumext } { wrong-miniright-place }
1502     }
1503     % outside
1504     \bool_lazy_and:nnT
1505     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
1506     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
1507     {
1508         \msg_error:nnn { enumext } { wrong-miniright-place }
1509     }
1510     % starred env
1511     \bool_if:NT \l__enumext_starred_bool
1512     {

```

```

1513     \msg_error:nnn { enumext } { wrong-miniright-starred }
1514   }
1515   \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
1516   {
1517     \__enumext_keyans_mini_right_cmd:n {#1}
1518   }
1519   { \__enumext_mini_right_cmd:n {#1} }
1520 }

```

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

`__enumext_mini_right_cmd:n`

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

```

1521 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1522 {
1523   \dim_compare:nNnTF
1524   { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1525   {
1526     \__enumext_multicols_stop:
1527     \int_compare:nNnTF
1528     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } = { 1 }
1529     {
1530       \par\addvspace{ \l__enumext_minipage_after_skip }
1531     }
1532     \end__enumext_mini_page
1533     \hfill
1534     \__enumext_mini_page{ \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } }
1535     \par\nointerlineskip
1536     \addvspace { \l__enumext_minipage_right_skip }
1537     \bool_if:nF {#1}
1538     {
1539       \centering
1540     }
1541     \int_gzero:N \g__enumext_minipage_stat_int
1542   }
1543   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1544   % paranoia
1545   \RenewDocumentCommand \miniright { s }
1546   {
1547     \msg_error:nn { enumext } { many-miniright-used }
1548   }
1549 }

```

(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.

```

1550 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1551 {
1552   \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1553   {
1554     \__enumext_keyans_multicols_stop:
1555     \int_compare:nNnTF { \l__enumext_columns_v_int } = { 1 }
1556     {
1557       \par\addvspace{ \l__enumext_minipage_after_skip }
1558     }
1559     \end__enumext_mini_page
1560     \hfill
1561     \__enumext_mini_page{ \l__enumext_minipage_right_v_dim }
1562     \par\nointerlineskip
1563     \addvspace { \l__enumext_minipage_right_skip }
1564     \bool_if:nF {#1}
1565     {
1566       \centering
1567     }
1568     \int_gzero:N \g__enumext_minipage_stat_int

```



```

1569     }
1570     { \msg_error:nnn { enumext } { wrong-miniright-use } }
1571 % paranoia
1572 \RenewDocumentCommand \miniright { s }
1573 {
1574     \msg_error:nn { enumext } { many-miniright-used }
1575 }
1576 }

```

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

12.23 Setting above and below keys

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

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

```

above* 1577 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
below 1578 {
below* 1579     \keys_define:nn { enumext / #1 }
1580     {
1581         above .skip_set:c = { \__enumext_vspace_above_#2_skip },
1582         above .value_required:n = true,
1583         above* .code:n = \bool_set_true:c { \__enumext_vspace_a_star_#2_bool }
1584                 \keys_set:nn { enumext / #1 } { above = {##1} },
1585         above* .value_required:n = true,
1586         below .skip_set:c = { \__enumext_vspace_below_#2_skip },
1587         below .value_required:n = true,
1588         below* .code:n = \bool_set_true:c { \__enumext_vspace_b_star_#2_bool }
1589                 \keys_set:nn { enumext / #1 } { below = {##1} },
1590         below* .value_required:n = true,
1591     }
1592 }
1593 \clist_map_inline:Nn \__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

12.23.1 Functions for above and below keys in enumext

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

```

1594 \cs_new_protected:Nn \__enumext_vspace_above:
1595 {
1596     \skip_if_eq:nnF
1597     { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1598     {
1599         \bool_if:cTF { \__enumext_vspace_a_star_ \__enumext_level: _bool }
1600         {
1601             \vspace*{ \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1602         }
1603         {
1604             \vspace { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1605         }
1606     }
1607 }

```

(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.

```

1608 \cs_new_protected:Nn \__enumext_vspace_below:
1609 {
1610     \skip_if_eq:nnF
1611     { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1612     {
1613         \bool_if:cTF { \__enumext_vspace_b_star_ \__enumext_level: _bool }
1614         {
1615             \vspace*{ \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1616         }

```

```

1617         {
1618             \vspace { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1619         }
1620     }
1621 }

```

(End of definition for __enumext_vspace_below:.)

12.23.2 Functions for above and below keys in keyans

__enumext_vspace_above_v: The function __enumext_vspace_above_v: apply the *vertical space above* the **keyans** environment set by the *above* and *above** keys.

```

1622 \cs_new_protected:Nn \__enumext_vspace_above_v:
1623 {
1624     \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1625     {
1626         \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1627         {
1628             \vspace*{ \l__enumext_vspace_above_v_skip }
1629         }
1630         { \vspace { \l__enumext_vspace_above_v_skip } }
1631     }
1632 }

```

(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.

```

1633 \cs_new_protected:Nn \__enumext_vspace_below_v:
1634 {
1635     \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1636     {
1637         \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1638         {
1639             \vspace*{ \l__enumext_vspace_below_v_skip }
1640         }
1641         { \vspace { \l__enumext_vspace_below_v_skip } }
1642     }
1643 }

```

(End of definition for __enumext_vspace_below_v:.)

12.23.3 Functions for above and below keys in enumext* keyans*

__enumext_vspace_above_vii: The functions __enumext_vspace_above_vii: and __enumext_vspace_above_viii: apply the *vertical space above* the **enumext*** and **keyans*** environments set by the *above* and *above** keys.

__enumext_vspace_above_viii:

```

1644 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1645 {
1646     \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1647     {
1648         \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1649         {
1650             \vspace*{ \l__enumext_vspace_above_vii_skip }
1651         }
1652         { \vspace { \l__enumext_vspace_above_vii_skip } }
1653     }
1654 }
1655 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1656 {
1657     \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1658     {
1659         \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1660         {
1661             \vspace*{ \l__enumext_vspace_above_viii_skip }
1662         }
1663         { \vspace { \l__enumext_vspace_above_viii_skip } }
1664     }
1665 }

```

(End of definition for __enumext_vspace_above_vii: and __enumext_vspace_above_viii:.)

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

`__enumext_vspace_below_viii:`

```

1666 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1667 {
1668   \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1669   {
1670     \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1671     {
1672       \vspace*{ \l__enumext_vspace_below_vii_skip }
1673     }
1674     { \vspace { \l__enumext_vspace_below_vii_skip } }
1675   }
1676 }
1677 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1678 {
1679   \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1680   {
1681     \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1682     {
1683       \vspace*{ \l__enumext_vspace_below_viii_skip }
1684     }
1685     { \vspace { \l__enumext_vspace_below_viii_skip } }
1686   }
1687 }

```

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

12.24 Setting series, resume and resume* keys

The `series` key is responsible for the whole process of the `resume` and `resume*` keys. The idea behind this is to be able to absorb the `<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*`.

```

1688 \cs_set_protected:Npn \__enumext_tmp:n #1
1689 {
1690   \keys_define:nn { enumext / #1 }
1691   {
1692     series .str_set:N = \l__enumext_series_str,
1693     series .value_required:n = true,
1694     resume .code:n = \__enumext_resume_series:n {##1},
1695     resume* .code:n = \__enumext_resume_starred:,
1696     resume* .value_forbidden:n = true,
1697   }
1698 }
1699 \clist_map_inline:nn { level-1, enumext* } { { \__enumext_tmp:n {#1} } }

```

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

12.24.1 Internal functions for series key

`__enumext_filter_series:n` 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.

`__enumext_filter_series_key:n`

`__enumext_filter_series_pair:nn`

```

1700 \cs_new:Npn \__enumext_filter_series:n #1
1701 {
1702   \use:e
1703   {
1704     \keyval_parse:NNn
1705     \__enumext_filter_series_key:n
1706     \__enumext_filter_series_pair:nn {#1}
1707   }
1708 }

```

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.

```

1709 \cs_new:Npn \__enumext_filter_series_key:n #1
1710 {
1711   \str_case:nnF {#1}
1712   {
1713     { resume } {} { resume* } {} { base-fix } {}
1714   }
1715   { , { \exp_not:n {#1} } }
1716 }

```

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

```

1717 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1718 {
1719   \str_case:nnF {#1}
1720   {
1721     { series } {} { resume } {} { start } {}
1722     { start* } {} { save-ans } {} { save-key } {}
1723   }
1724   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1725 }

```

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

```

\__enumext_parse_series:n
\__enumext_resume_last:n

```

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

- The value of boolean variable `\l__enumext_resume_active_bool` is set to true by the function `__enumext_resume_counter:n` which is used by the keys `resume` and `resume*`, in this case we must Make sure it is set to false so that it does not overwrite the default filtered *(keys)*. This function is passed to the function `__enumext_parse_keys:n` in the `enumext` environment definition (§12.38) and to the function `__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§12.43).

```

1726 \cs_new_protected:Npn \__enumext_parse_series:n #1
1727 {
1728   \str_if_empty:NTF \l__enumext_series_str
1729   {
1730     \bool_if:NF \l__enumext_resume_active_bool
1731     {
1732       \__enumext_resume_last:n {#1}
1733     }
1734   }
1735   {
1736     \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str_tl }
1737     \tl_gset:ce { g__enumext_series_ \l__enumext_series_str_tl }
1738     { \__enumext_filter_series:n {#1} }
1739     \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str_int }
1740     {
1741       \int_new:c { g__enumext_series_ \l__enumext_series_str_int }
1742     }
1743   }
1744 }

```

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.

```

1745 \cs_new_protected:Npn \__enumext_resume_last:n #1
1746 {
1747   \bool_if:NT \l__enumext_standar_first_bool
1748   {
1749     \tl_gclear:N \g__enumext_standar_series_tl
1750     \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1751   }
1752   \bool_if:NT \l__enumext_starred_first_bool
1753   {
1754     \tl_gclear:N \g__enumext_starred_series_tl
1755     \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1756   }
1757 }

```

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

12.24.2 Internal function to save counter value

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

- The variables `\l__enumext_series_str` and `\l__enumext__resume_name_tl` contain the same `{⟨series name⟩}` but are executed at different moments, the integer variable with `\l__enumext_series_str` sets the value when execute `series={⟨series name⟩}` and the integer variable with `\l__enumext__resume_name_tl` sets the subsequent values when use `resume={⟨series name⟩}`. This function is passed to the `enumext` environment definition (§12.38) and the `enumext*` environment definition (§12.43).

```

1758 \cs_new_protected:Nn \__enumext_resume_save_counter:
1759 {
1760   \bool_if:NT \g__enumext_standar_bool
1761   {
1762     \tl_if_empty:NF \l__enumext_series_str
1763     {
1764       \int_gset_eq:cN
1765       { g__enumext_series_ \l__enumext_series_str_int } \value{enumXi}
1766     }
1767     \tl_if_empty:NTF \l__enumext_resume_name_tl
1768     {
1769       \str_if_empty:NT \l__enumext_series_str
1770       {
1771         \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1772       }
1773     }
1774     {
1775       \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl_int }
1776       {
1777         \int_gset_eq:cN
1778         { g__enumext_series_ \l__enumext_resume_name_tl_int } \value{enumXi}
1779       }
1780     }
1781     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl_int }
1782     {
1783       \int_gset_eq:cN
1784       { g__enumext_resume_ \l__enumext_store_name_tl_int } \value{enumXi}
1785     }
1786   }
1787   \bool_if:NT \g__enumext_starred_bool
1788   {
1789     \tl_if_empty:NF \l__enumext_series_str
1790     {
1791       \int_gset_eq:cN
1792       { g__enumext_series_ \l__enumext_series_str_int } \value{enumXvii}
1793     }
1794     \tl_if_empty:NTF \l__enumext_resume_name_tl
1795     {
1796       \str_if_empty:NT \l__enumext_series_str
1797       {
1798         \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1799       }
1800     }
1801     {
1802       \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl_int }
1803       {
1804         \int_gset_eq:cN
1805         { g__enumext_series_ \l__enumext_resume_name_tl_int } \value{enumXvii}
1806       }
1807     }
1808     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl_int }
1809     {
1810       \int_gset_eq:cN
1811       { g__enumext_resume_ \l__enumext_store_name_tl_int } \value{enumXvii}
1812     }
1813   }
1814 }

```

(End of definition for `__enumext_resume_save_counter:`.)

12.24.3 Internal functions for resume key

`__enumext_resume_series:n`

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

```

1815 \cs_new_protected:Npn \__enumext_resume_series:n #1
1816 {
1817   \tl_if_empty:nTF {#1}
1818   {
1819     \__enumext_resume_counter:n { }
1820   }
1821   {
1822     \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1823     {
1824       \__enumext_resume_counter:n {#1}
1825       \bool_if:NT \g__enumext_standar_bool
1826       {
1827         \keys_set:nv { enumext / level-1 }
1828         { g__enumext_series_ \tl_to_str:n {#1} _tl }
1829       }
1830       \bool_if:NT \g__enumext_starred_bool
1831       {
1832         \keys_set:nv { enumext / enumext* }
1833         { g__enumext_series_ \tl_to_str:n {#1} _tl }
1834       }
1835     }
1836     {
1837       \bool_if:NT \g__enumext_standar_bool
1838       {
1839         \msg_error:nnn { enumext } { unknown-series } {#1}
1840       }
1841       \bool_if:NT \g__enumext_starred_bool
1842       {
1843         \msg_error:nnn { enumext } { unknown-series } {#1}
1844       }
1845     }
1846   }
1847 }

```

(End of definition for `__enumext_resume_series:n`)

`__enumext_resume_counter:n`

`__enumext_resume_counter:`

`__enumext_resume_counter_series:`

`__enumext_resume_counter_save_ans:`

The function `__enumext_resume_counter:n` will set the variable `\l__enumext_resume_active_bool` to true and pass the value of the key `resume` to the variable `\l__enumext_series_name_tl` which will contain the `{⟨series name⟩}`. If the variable `\l__enumext_series_name_tl` is empty, that is, we are passing the key `resume without value`, we will execute the function `__enumext_resume_counter:`; otherwise, when we pass `resume={⟨series name⟩}` we will execute the function `__enumext_resume_counter_series:`, finally we will execute the function `__enumext_resume_counter_save_ans:` which is associated with the key `save-ans`.

```

1848 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1849 {
1850   \bool_set_true:N \l__enumext_resume_active_bool
1851   \tl_set:Nn \l__enumext_resume_name_tl {#1}
1852   \tl_if_empty:NTF \l__enumext_resume_name_tl
1853   {
1854     \__enumext_resume_counter:
1855   }
1856   {
1857     \__enumext_resume_counter_series:
1858   }
1859   \__enumext_resume_counter_save_ans:
1860 }

```

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.

```

1861 \cs_new_protected:Nn \__enumext_resume_counter:
1862 {
1863   \bool_if:NT \g__enumext_standar_bool

```



```

1864     {
1865         \int_gincr:N \g__enumext_resume_int
1866         \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1867     }
1868     \bool_if:NT \g__enumext_starred_bool
1869     {
1870         \int_gincr:N \g__enumext_resume_vii_int
1871         \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1872     }
1873 }

```

The function `__enumext_resume_counter_series:` will be executed when the `resume={⟨series name⟩}` key is active, setting the counters for the “*first level*” of the environments according to the value of the integer variables created by the `series` key.

```

1874 \cs_new_protected:Nn \__enumext_resume_counter_series:
1875 {
1876     \bool_if:NT \g__enumext_standar_bool
1877     {
1878         \int_set:Nn \l__enumext_start_i_int
1879         {
1880             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1881         }
1882     }
1883     \bool_if:NT \g__enumext_starred_bool
1884     {
1885         \int_set:Nn \l__enumext_start_vii_int
1886         {
1887             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1888         }
1889     }
1890 }

```

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.

```

1891 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1892 {
1893     \bool_lazy_and:nnT
1894     { \bool_if_p:N \l__enumext_standar_first_bool }
1895     { \bool_if_p:N \l__enumext_store_active_bool }
1896     {
1897         \int_set:Nn \l__enumext_start_i_int
1898         {
1899             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1900         }
1901     }
1902     \bool_lazy_and:nnT
1903     { \bool_if_p:N \l__enumext_starred_first_bool }
1904     { \bool_if_p:N \l__enumext_store_active_bool }
1905     {
1906         \int_set:Nn \l__enumext_start_vii_int
1907         {
1908             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1909         }
1910     }
1911 }

```

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

12.24.4 Internal function for `resume*` key

`__enumext_resume_starred:`

The function `__enumext_resume_starred:` will handle the `resume*` key in the `enumext` and `enumext*` environments. This function will execute the filtered `⟨keys⟩` in the last one and will continue with the numbering according to the last execution of the environment `enumext` or `enumext*` in which the keys `resume={⟨series name⟩}` or `series={⟨series name⟩}` were not active.

```

1912 \cs_new_protected:Nn \__enumext_resume_starred:
1913 {
1914     \bool_if:NT \g__enumext_standar_bool
1915     {
1916         \tl_if_empty:NF \g__enumext_standar_series_tl
1917         {
1918             \__enumext_resume_counter:n { }

```

```

1919         \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1920     }
1921 }
1922 \bool_if:NT \g__enumext_starred_bool
1923 {
1924     \tl_if_empty:NF \g__enumext_starred_series_tl
1925     {
1926         \__enumext_resume_counter:n { }
1927         \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1928     }
1929 }
1930 }

```

(End of definition for __enumext_resume_starred:.)

12.25 Setting save-ans, check-ans and no-store keys

The key `save-ans` is directly associated with the keys `check-ans`, `no-store`, `resume` and `resume*`, this will activate the entire “storage system” in the `enumext` package.

12.25.1 Setting save-ans key

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

```

1931 \cs_set_protected:Npn \__enumext_tmp:n #1
1932 {
1933     \keys_define:nn { enumext / #1 }
1934     {
1935         save-ans .code:n = \__enumext_storing_set:n {##1},
1936         save-ans .value_required:n = true,
1937     }
1938 }
1939 \clist_map_inline:nn { level-1, enumext* } { { \__enumext_tmp:n {#1} } }

```

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

12.25.2 Internal functions for save-ans key

`__enumext_start_save_ans_msg:` The functions `__enumext_start_save_ans_msg:` and `__enumext_stop_save_ans_msg:` will display in the terminal and `.log` file the environment in which the `save-ans` key was executed along with the line at the beginning and end of it. The function `__enumext_start_save_ans_msg:` will be passed to `__enumext_storing_set:n` and the function `__enumext_stop_save_ans_msg:` will be passed to the function `__enumext_execute_after_env:`.

```

1940 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
1941 {
1942     \msg_term:nnVV { enumext } { save-ans-log }
1943     \g__enumext_envir_name_tl \l__enumext_store_name_tl
1944 }
1945 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
1946 {
1947     \msg_term:nnVV { enumext } { save-ans-log-hook }
1948     \g__enumext_envir_name_tl \g__enumext_store_name_tl
1949 }

```

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

`__enumext_storing_set:n` The function `__enumext_storing_set:n` first pass the value of the `save-ans` key to the variable `\l__enumext_store_name_tl` which will contain the “store name” of the *(sequence)* and *(prop list)* we will use. If `\l__enumext_store_name_tl` is *empty* we return an error message, otherwise will return the appropriate message `__enumext_start_save_ans_msg:` and proceed to execute the function `__enumext_storing_exec:` for `enumext` and `enumext*` environments.

```

1950 \cs_new_protected:Npn \__enumext_storing_set:n #1
1951 {
1952     \tl_set:Nx \l__enumext_store_name_tl {#1}
1953     \tl_if_empty:NTF \l__enumext_store_name_tl
1954     {
1955         \bool_lazy_or:nnT
1956         { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1957         {
1958             \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
1959         }
1960     }
1961     {
1962         \bool_lazy_or:nnT

```

```

1963         { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1964     {
1965         __enumext_start_save_ans_msg:
1966         __enumext_storing_exec:
1967     }
1968 }
1969 }

```

The function `__enumext_storing_exec:` will set to true the variable `\l__enumext_store_active_bool` which activates the use of the `\anskey` command and the `keyans`, `keyans*` and `keyanspic` environments and will set to true the variable `\l__enumext_check_answers_bool` used for checking answers by the `check-ans` and `no-store` keys, copy `{\store name}` into the global variable `\g__enumext_store_name_tl` and execute the function `__enumext_anskey_env_make:V` creating the environment `anskey*` (§12.30). The `\prop list` `\g__enumext_series_{store name}_prop` and the `\sequence` `\g__enumext_series_{store name}_seq` will be created globally to “store content” in case they do not exist together with the integer variable `\g__enumext_series_{store name}_int` used by the keys `resume` and `resume*`.

```

1970 \cs_new_protected:Nn \__enumext_storing_exec:
1971 {
1972     \bool_set_true:N \l__enumext_store_active_bool
1973     \bool_set_true:N \l__enumext_check_answers_bool
1974     \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1975     \__enumext_anskey_env_make:V \l__enumext_store_name_tl
1976     \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1977     {
1978         \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
1979         \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1980     }
1981     \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1982     {
1983         \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
1984         \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1985     }
1986     \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1987     {
1988         \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
1989         \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1990     }
1991 }

```

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

12.25.3 The check answer mechanism

The mechanism for checking that all questions are answered follows this logic:

If the line begins with `\item` or `\item*` and does NOT *open a nested environment*, each `\item` or `\item*` must contain a *single* execution of the `\anskey` command, i.e. the counter of the executions of the `\anskey` command must be equal to the counter associated with the sum of executions of `\item` and `\item*`.

If the line begins with `\item` or `\item*` and *opens a nested environment* each `\item` or `\item*` in the nested environment must have a *single* execution of the `\anskey` command and the counter associated to the sum of `\item` and `\item*` executions must decrementing by “one” to maintain equality.

In order for the mechanism for the check-answer to work (not counting `keyans`, `keyans*` and `keyanspic`) we need:

1. We must keep track of the total number of `\item` and `\item*` (enumerated) that appear within the environment including the nested levels.
2. We must keep track of the total number of `\item` and `\item*` (enumerated) that appear per level of nesting.
3. Keeping track of the number of times the environment nests.

The integer variable associated to the sum of each `\item` and `\item*` in the environment `\g__enumext_item_number_int` must match the integer variable `\g__enumext_item_anskey_int` associated to the execution of the command `\anskey`. We analyze the cases:

- a) If the list only has one level the number of `\item` + `\item*` = `\anskey`
- b) If the list has *nested levels*, for each level of nesting we need to decrementing by one (for the `\item` or `\item*` that opens the nest) so that the account remains the same.

With `keyans`, `keyans*` and `keyanspic` it is enough to increase in one the integer of `\anskey`. The integers created must be global if they are not lost in the interior levels of nesting and to execute the test we will use a “hook” function after closing the first level of the environment.

12.25.4 Setting check-ans and no-store keys

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

```

check-ans 1992 \cs_set_protected:Npn \__enumext_tmp:n #1
no-store 1993 {
1994   \keys_define:nn { enumext / #1 }
1995   {
1996     check-ans .bool_set:N = \__enumext_check_ans_key_bool,
1997     check-ans .initial:n = false,
1998     check-ans .value_required:n = true,
1999     no-store .code:n = {
2000       \bool_set_false:N \__enumext_check_answers_bool
2001       \bool_set_false:N \__enumext_check_ans_key_bool
2002     },
2003     no-store .value_forbidden:n = true,
2004   }
2005 }
2006 \clist_map_inline:nn
2007 {
2008   level-1, level-2, level-3, level-4, enumext*
2009 }
2010 { \__enumext_tmp:n {#1} }
```

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

12.25.5 Set-up check answer mechanism

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

```

2011 \cs_new_protected:Nn \__enumext_check_ans_active:
2012 {
2013   \tl_if_empty:NF \__enumext_store_name_tl
2014   {
2015     \bool_if:NT \__enumext_check_answers_bool
2016     {
2017       \__enumext_check_ans_level:
2018     }
2019   }
2020 }
```

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

```

2021 \cs_new_protected:Nn \__enumext_check_ans_level:
2022 {
2023   \int_case:nn { \__enumext_level_int }
2024   {
2025     { 1 }{
2026       \bool_lazy_all:nT
2027       {
2028         { \bool_if_p:N \g__enumext_starred_bool }
2029         { \int_compare_p:nNn { \__enumext_level_h_int } = { 1 } }
2030       }
2031       {
2032         \int_gdecr:N \g__enumext_item_number_int
2033         \bool_set_false:N \__enumext_item_number_bool
2034       }
2035     }
2036     { 2 }{
2037       \int_gdecr:N \g__enumext_item_number_int
2038       \bool_set_false:N \__enumext_item_number_bool
2039     }
2040     { 3 }{
2041       \int_gdecr:N \g__enumext_item_number_int
2042       \bool_set_false:N \__enumext_item_number_bool
2043     }
2044     { 4 }{
2045       \int_gdecr:N \g__enumext_item_number_int
2046       \bool_set_false:N \__enumext_item_number_bool

```

```

2047         }
2048     }

```

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.

```

2049     \int_case:nn { \l__enumext_level_h_int }
2050     {
2051         { 1 }{
2052             \bool_lazy_all:nT
2053             {
2054                 { \bool_if_p:N \g__enumext_standar_bool }
2055                 { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
2056             }
2057             {
2058                 \int_gdecr:N \g__enumext_item_number_int
2059                 \bool_set_false:N \l__enumext_item_number_bool
2060             }
2061         }
2062     }
2063 }

```

(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.

```

2064 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
2065 {
2066     \bool_lazy_and:nnT
2067     { \bool_if_p:N \l__enumext_check_ans_key_bool }
2068     { \bool_if_p:N \g__enumext_standar_bool }
2069     {
2070         \bool_gset_true:N \g__enumext_check_ans_key_bool
2071     }
2072     \bool_lazy_and:nnT
2073     { \bool_if_p:N \l__enumext_check_ans_key_bool }
2074     { \bool_if_p:N \g__enumext_starred_bool }
2075     {
2076         \bool_gset_true:N \g__enumext_check_ans_key_bool
2077     }
2078 }

```

(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:`.

```

2079 \cs_new_protected:Nn \__enumext_item_answer_diff:
2080 {
2081     \int_gset:Nn \g__enumext_item_answer_diff_int
2082     {
2083         \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
2084     }
2085 }

```

(End of definition for `__enumext_item_answer_diff:`.)

`__enumext_check_ans_show:`

`__enumext_check_ans_msg_less:`
`__enumext_check_ans_msg_same_ok:`
`__enumext_check_ans_msg_greater:`

The function `__enumext_check_ans_show:` will be executed within the function `__enumext_execute_after_env:` when the key `check-ans` is active, that is, when `\g__enumext_check_ans_key_bool` is “*true*” and will return the appropriate message according to the value of `\g__enumext_item_answer_diff_int` set by the function `__enumext_item_answer_diff:`.

```

2086 \cs_new_protected:Nn \__enumext_check_ans_show:
2087 {
2088     \int_case:nn { \g__enumext_item_answer_diff_int }
2089     {
2090         { -1 }{ \__enumext_check_ans_msg_less: }
2091         { 0 }{ \__enumext_check_ans_msg_same_ok: }
2092         { 1 }{ \__enumext_check_ans_msg_greater: }
2093     }
2094 }

```

```

2095 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
2096 {
2097   \msg_warning:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2098   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2099 }
2100 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
2101 {
2102   \msg_term:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2103   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2104 }
2105 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
2106 {
2107   \msg_warning:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2108   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2109 }

```

(End of definition for __enumext_check_ans_show: and others.)

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:`.

```

2110 \cs_new_protected:Nn \__enumext_check_ans_log:
2111 {
2112   \int_case:nn { \g__enumext_item_answer_diff_int }
2113   {
2114     { -1 } { \__enumext_check_ans_log_msg_less: }
2115     { 0 } { \__enumext_check_ans_log_msg_same_ok: }
2116     { 1 } { \__enumext_check_ans_log_msg_greater: }
2117   }
2118 }
2119 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
2120 {
2121   \msg_log:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2122   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2123 }
2124 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
2125 {
2126   \msg_log:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2127   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2128 }
2129 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
2130 {
2131   \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2132   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2133 }

```

(End of definition for __enumext_check_ans_log: and others.)

12.25.6 Check for \item* and \anspic* commands

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.

```

2134 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
2135 {
2136   \int_compare:nNnT
2137   { \g__enumext_check_starred_cmd_int } = { 0 }
2138   {
2139     \msg_warning:nnnV
2140     { enumext } { missing-starred } { #1 } \l__enumext_check_start_line_env_tl
2141   }
2142   \int_compare:nNnT
2143   { \g__enumext_check_starred_cmd_int } > { 1 }
2144   {
2145     \msg_warning:nnnV
2146     { enumext } { many-starred } { #1 } \l__enumext_check_start_line_env_tl
2147   }
2148   \int_gzero:N \g__enumext_check_starred_cmd_int
2149   \tl_clear:N \l__enumext_check_start_line_env_tl
2150 }

```

(End of definition for __enumext_check_starred_cmd:n.)

12.26 Keys and functions associated with storage

We add the keys `wrap-ans`, `wrap-opt`, `save-sep`, `mark-ans`, `mark-pos`, `show-ans`, `show-pos`, `mark-ref` and `save-ref` related to the “*storage system*” and internal mechanism of “*label and ref*” only at the *first level* of `enumext` and `enumext*`.

```

2151 \cs_set_protected:Npn \__enumext_tmp:n #1
2152 {
2153   \keys_define:nn { enumext / #1 }
2154   {
2155     wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
2156     wrap-ans .initial:n =
2157       {
2158         \fbox{\parbox[t]{\dimeval{\itemwidth -2\fboxsep -2\fboxrule}}{##1}}
2159       },
2160     wrap-ans .value_required:n = true,
2161     wrap-opt .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
2162     wrap-opt .initial:n = [{##1}],
2163     wrap-opt .value_required:n = true,
2164     save-sep .tl_set:N = \__enumext_store_keyans_item_opt_sep_tl,
2165     save-sep .initial:n = {, ~ },
2166     save-sep .value_required:n = true,
2167     mark-ans .tl_set:N = \__enumext_mark_answer_sym_tl,
2168     mark-ans .initial:n = \textasteriskcentered,
2169     mark-ans .value_required:n = true,
2170     mark-pos .choice:,
2171     mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
2172     mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
2173     mark-pos / unknown .code:n =
2174       \msg_error:nneee { enumext } { unknown-choice }
2175       { mark-pos } { left, ~ right } { \exp_not:n {##1} },
2176     mark-pos .initial:n = right,
2177     mark-pos .value_required:n = true,
2178     show-ans .bool_set:N = \__enumext_show_answer_bool,
2179     show-ans .initial:n = false,
2180     show-ans .value_required:n = true,
2181     show-pos .bool_set:N = \__enumext_show_position_bool,
2182     show-pos .initial:n = false,
2183     show-pos .value_required:n = true,
2184     mark-ref .tl_set:N = \__enumext_mark_ref_sym_tl,
2185     mark-ref .initial:n = \textasteriskcentered,
2186     mark-ref .value_required:n = true,
2187     save-ref .bool_set:N = \__enumext_store_ref_key_bool,
2188     save-ref .initial:n = false,
2189     save-ref .value_required:n = true,
2190   }
2191 }
2192 \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`.

```

2193 \cs_set_protected:Npn \__enumext_tmp:n #1
2194 {
2195   \keys_define:nn { enumext / #1 }
2196   {
2197     mark-pos .choice:,
2198     mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
2199     mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
2200     mark-pos .initial:n = right,
2201     mark-pos .value_required:n = true,
2202     show-ans .bool_set:N = \__enumext_show_answer_bool,
2203     show-ans .initial:n = false,
2204     show-ans .value_required:n = true,
2205     show-pos .bool_set:N = \__enumext_show_position_bool,
2206     show-pos .initial:n = false,
2207     show-pos .value_required:n = true,
2208   }
2209 }
2210 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }

```

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

12.26.1 Store optional arguments of the environments

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

```

__enumext_store_active_keys:n
__enumext_store_active_keys_vii:n

```

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

```

2211 \cs_new_protected:Npn __enumext_store_active_keys:n #1
2212 {
2213   \bool_if:cF { __enumext_store_save_key_ __enumext_level: _bool }
2214   {
2215     \tl_clear:c { __enumext_save_key_ __enumext_level: _tl }
2216     \tl_set:ce
2217       { __enumext_store_save_key_ __enumext_level: _tl }
2218       { __enumext_filter_save_key:n {#1} }
2219   }
2220 }
2221 \cs_new_protected:Npn __enumext_store_active_keys_vii:n #1
2222 {
2223   \bool_if:NF __enumext_store_save_key_vii_bool
2224   {
2225     \tl_clear:N __enumext_store_save_key_vii_tl
2226     \tl_set:Ne __enumext_store_save_key_vii_tl { __enumext_filter_save_key:n {#1} }
2227   }
2228 }

```

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

12.26.2 Setting save-key key

Since this list structure will be stored in the *(sequence)* established by the `save-ans` key when executing `\anskey`, we will not be able to modify it. The best thing here is to have a key that allows you to modify the optional argument of the list stored in the *(sequence)*.

`save-key`

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

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

```

2229 \cs_set_protected:Npn __enumext_tmp:n #1
2230 {
2231   \keys_define:nn { enumext / enumext* }
2232   {
2233     save-key .code:n = __enumext_parse_save_key_vii:n {##1},
2234     save-key .value_required:n = true,
2235   }
2236   \keys_define:nn { enumext / #1 }
2237   {
2238     save-key .code:n = __enumext_parse_save_key:n {##1},
2239     save-key .value_required:n = true,
2240   }
2241 }
2242 \clist_map_inline:nn { level-1, level-2, level-3, level-4 } { __enumext_tmp:n {#1} }

```

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

```

__enumext_parse_save_key:n
__enumext_parse_save_key_vii:n

```

The functions `__enumext_parse_save_key:n` and `__enumext_parse_save_key_vii:n` will be responsible for storing the filtered *(keys)* in the variable `__enumext_store_save_key_X_tl` for `enumext` and `enumext*`.

```

2243 \cs_new_protected:Npn __enumext_parse_save_key:n #1
2244 {
2245   \bool_set_true:c { __enumext_store_save_key_ __enumext_level: _bool }
2246   \tl_clear:c { __enumext_save_key_ __enumext_level: _tl }
2247   \tl_set:ce
2248     { __enumext_store_save_key_ __enumext_level: _tl }
2249     { __enumext_filter_save_key:n {#1} }
2250 }

```

```

2251 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2252 {
2253   \bool_set_true:N \l__enumext_store_save_key_vii_bool
2254   \tl_clear:N \l__enumext_store_save_key_vii_tl
2255   \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2256 }

```

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

12.26.3 Internal functions to store optional arguments

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

```

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

```

```

2257 \cs_new:Npn \__enumext_filter_save_key:n #1
2258 {
2259   \use:e
2260   {
2261     \keyval_parse:NNn
2262     \__enumext_filter_save_key_key:n
2263     \__enumext_filter_save_key_pair:nn {#1}
2264   }
2265 }

```

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.

```

2266 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2267 {
2268   \str_case:nnF {#1}
2269   {
2270     { resume } {} { resume* } {} { no-store } {} { base-fix } {}
2271   }
2272   { , { \exp_not:n {#1} } }
2273 }

```

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.

```

2274 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2275 {
2276   \str_case:nnF {#1}
2277   {
2278     { series } {} { resume } {} { save-ans } {} { save-ref } {}
2279     { save-key } {} { check-ans } {} { show-ans } {} { show-pos } {}
2280     { wrap-ans } {} { mark-ans } {} { wrap-opt } {} { save-sep } {}
2281     { mark-ref } {} { mini-env } {} { mini-sep } {} { mini-right } {}
2282     { mini-right* } {}
2283   }
2284   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2285 }

```

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

12.26.4 Function for storing content in prop list

```

\__enumext_store_addto_prop:n
\__enumext_store_addto_prop:V

```

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

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

```

2286 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2287 {
2288   \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2289   {
2290     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2291   }
2292   { #1 }
2293 }
2294 \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V, e }

```

(End of definition for __enumext_store_addto_prop:n.)

12.26.5 Function for storing content in sequence

The function `__enumext_store_addto_seq:n` stores the content in *⟨sequence⟩* defined by `save-ans` key. This function is used by `\anskey` in `enumext`, `\item*` in `keyans` and `\anspic` in `keyanspic`. The form in which the content is stored in *⟨sequence⟩* is in an internal `enumext` or `enumext*` environments with the *same structure* in which the command was executed. The “*stored content*” is retrieved by means of the `\printkeyans` command.

```

2295 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2296 {
2297   \seq_gput_right:cn { g__enumext_ \__enumext_store_name_tl_seq } { #1 }
2298 }
2299 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V, e }

```

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

12.26.6 Functions for storing the list structure in the sequence

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

```

2300 \cs_new_protected:Nn \__enumext_store_level_open:
2301 {
2302   \bool_if:NT \__enumext_check_answers_bool
2303   {
2304     \tl_if_empty:cTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2305     {
2306       \__enumext_store_addto_seq:n
2307       {
2308         \item \begin{enumext}
2309       }
2310     }
2311     {
2312       \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2313       {
2314         \item \begin{enumext} [
2315       }
2316       \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2317       {
2318         ]
2319       }
2320       \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2321     }
2322   }
2323 }
2324 \cs_new_protected:Nn \__enumext_store_level_close:
2325 {
2326   \bool_if:NT \__enumext_check_answers_bool
2327   {
2328     \__enumext_store_addto_seq:n { \end{enumext} }
2329   }
2330 }

```

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

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

```

2331 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2332 {
2333   \bool_if:NT \__enumext_check_answers_bool
2334   {
2335     \tl_if_empty:NTF l__enumext_store_save_key_vii_tl
2336     {
2337       \__enumext_store_addto_seq:n
2338       {
2339         \item \begin{enumext*}
2340       }
2341     }
2342     {
2343       \tl_put_left:Nn l__enumext_store_save_key_vii_tl
2344       {
2345         \item \begin{enumext*}[
2346       }
2347       \tl_put_right:Nn l__enumext_store_save_key_vii_tl

```

```

2348         {
2349         }
2350     }
2351     \__enumext_store_addto_seq:V \__enumext_store_save_key_vii_tl
2352 }
2353 }
2354 }
2355 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2356 {
2357     \bool_if:NT \__enumext_check_answers_bool
2358     {
2359         \__enumext_store_addto_seq:n { \end{enumext*} }
2360     }
2361 }

```

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

12.26.7 Function for show marks and position

__enumext_print_keyans_box:NN
__enumext_print_keyans_box:cc

The function __enumext_print_keyans_box:NN print a box in the left margin with \l__enumext_mark_answer_sym_tl used by the `wrap-ans`, `show-ans` and `show-pos` keys. The function takes two arguments:

#1: \l__enumext_labelwidth_X_dim
#2: \l__enumext_labelsep_X_dim

```

2362 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2363 {
2364     \mode_leave_vertical:
2365     \skip_horizontal:n { -\dim_use:N #2 }
2366     \makebox[0pt][ r ]
2367     {
2368         \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2369         {
2370             \tl_use:N \l__enumext_mark_answer_sym_tl
2371         }
2372     }
2373     \skip_horizontal:n { \dim_use:N #2 }
2374 }
2375 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }

```

(End of definition for __enumext_print_keyans_box:NN.)

12.27 The internal label and ref

The function __enumext_store_internal_ref: handles the internal “label and ref” system used by the `save-ref` and `mark-ref` keys for \anskey will allow to execute \ref{⟨store name : position⟩} and will return 1.(a).i.A.

__enumext_store_internal_ref:

First we will remove the dots “.” from the current ⟨labels⟩, we do not want to get double dots in our references, then we will place this in the variable \l__enumext_newlabel_arg_two_tl.

```

2376 \cs_new_protected:Nn \__enumext_store_internal_ref:
2377 {
2378     \cs_set_protected:Npn \__enumext_tmp:n ##1
2379     {
2380         \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2381         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2382         \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2383         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2384     }
2385     \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2386     \cs_set:Npn \__enumext_tmp:n ##1
2387     { . \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.

```

2388     \bool_lazy_all:nT
2389     {
2390         { \bool_if_p:N \g__enumext_starred_bool }
2391         { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2392     }
2393     {
2394         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2395         { \tl_use:N \l__enumext_label_copy_vii_tl }
2396     }

```

```

2397 \bool_lazy_all:nT
2398 {
2399   { \bool_not_p:n { \g__enumext_standar_bool } }
2400   { \bool_if_p:N \l__enumext_standar_bool }
2401   { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } } }
2402 }
2403 {
2404   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2405   {
2406     \tl_use:N \l__enumext_label_copy_vii_tl
2407     \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2408   }
2409 }

```

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.

```

2410 \bool_lazy_all:nT
2411 {
2412   { \bool_if_p:N \g__enumext_standar_bool }
2413   { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } } }
2414 { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } } }
2415 }
2416 {
2417   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2418   {
2419     \tl_use:N \l__enumext_label_copy_i_tl
2420     \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2421   }
2422 }
2423 \cs_set:Npn \__enumext_tmp:n ##1
2424 { \tl_use:c { \l__enumext_label_copy_ \int_to_roman:n {##1} _tl } . }
2425 \bool_lazy_all:nT
2426 {
2427   { \bool_if_p:N \g__enumext_standar_bool }
2428   { \bool_if_p:N \l__enumext_starred_bool }
2429   { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } } }
2430 }
2431 {
2432   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2433   {
2434     \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2435     \tl_use:N \l__enumext_label_copy_vii_tl
2436   }
2437 }

```

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

```

2438 \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2439 {
2440   \l__enumext_store_name_tl \c_colon_str
2441   \int_eval:n { \prop_count:c { \g__enumext_ \l__enumext_store_name_tl _prop } }
2442 }

```

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

```

2443 \tl_put_right:Ne \l__enumext_write_aux_file_tl
2444 {
2445   \__enumext_newlabel:nn
2446   { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2447   { \l__enumext_newlabel_arg_two_tl }
2448 }
2449 \l__enumext_write_aux_file_tl
2450 }

```

(End of definition for `__enumext_store_internal_ref:.`)

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

`__enumext_store_anskey_code:n`

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

```

2451 \cs_new_protected:Npn \__enumext_store_anskey_code:n #1
2452 {

```

```

2453 \int_gincr:N \g__enumext_item_anskey_int
2454 \__enumext_store_addto_prop:n {#1}
2455 \bool_if:NT \l__enumext_store_ref_key_bool
2456 {
2457   \__enumext_store_internal_ref:
2458 }
2459 \__enumext_anskey_show_wrap_left:n { #1 }

```

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

```

2460 \tl_clear:N \l__enumext_store_anskey_arg_tl
2461 \bool_lazy_and:nnT
2462 { \bool_if_p:N \l__enumext_store_columns_break_bool }
2463 { \bool_not_p:n { \l__enumext_starred_bool } }
2464 {
2465   \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2466 }
2467 \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { \item }

```

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

```

2468 \bool_lazy_and:nnT
2469 { \bool_not_p:n { \l__enumext_starred_bool } }
2470 { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2471 {
2472   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2473   {
2474     ( \exp_not:V \l__enumext_store_item_join_int )
2475   }
2476 }

```

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

```

2477 \bool_if:NTF \l__enumext_store_item_star_bool
2478 {
2479   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2480   \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2481   {
2482     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2483     {
2484       [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2485     }
2486   }
2487   \dim_compare:nT
2488   {
2489     \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2490   }
2491   {
2492     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2493     {
2494       [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2495     }
2496   }
2497   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2498 }
2499 {
2500   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2501 }

```

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

```

2502 \bool_lazy_and:nnT
2503 { \bool_if_p:N \l__enumext_store_ref_key_bool }
2504 { \bool_if_p:N \l__enumext_hyperref_bool }
2505 {
2506   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2507   {
2508     \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2509     { \exp_not:V \l__enumext_mark_ref_sym_tl }
2510   }

```

```

2511     }
2512     \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2513 }

```

(End of definition for __enumext_store_anskey_code:n.)

__enumext_anskey_show_wrap_arg:n

The function __enumext_anskey_show_wrap_arg:n “wraps” the $\langle argument \rangle$ passed to \anskey and the $\langle body \rangle$ for anskey* when using the wrap-ans key.

```

2514 \cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
2515 {
2516   \par
2517   \bool_if:NTF \l__enumext_starred_bool
2518   {
2519     \__enumext_print_keyans_box:NN \l__enumext_labelwidth_vii_dim \l__enumext_labelsep_vii_dim
2520   }
2521   {
2522     \__enumext_print_keyans_box:cc
2523     { \l__enumext_labelwidth_ \l__enumext_level: _dim }
2524     { \l__enumext_labelsep_ \l__enumext_level: _dim }
2525   }
2526   \__enumext_anskey_wrapper:n { #1 }
2527 }

```

(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.

```

2528 \cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
2529 {
2530   \bool_if:NT \l__enumext_show_answer_bool
2531   {
2532     \__enumext_anskey_show_wrap_arg:n { #1 }
2533   }
2534   \bool_if:NT \l__enumext_show_position_bool
2535   {
2536     \tl_set:Nx \l__enumext_mark_answer_sym_tl
2537     {
2538       \group_begin:
2539       \exp_not:N \normalfont
2540       \exp_not:N \footnotesize [ \int_eval:n
2541       {
2542         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2543       }
2544       ]
2545       \group_end:
2546     }
2547     \__enumext_anskey_show_wrap_arg:n { #1 }
2548   }
2549 }

```

(End of definition for __enumext_anskey_show_wrap_left:n.)

12.29 The command \anskey

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

The \anskey command will cover this point and give it similar behaviour to that of \item in the enumext and enumext* environments executed as follows \anskey[$\langle key = val \rangle$]{ $\langle content \rangle$ }.

__enumext_anskey_unknown:n

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

__enumext_anskey_unknown:nn

```

2550 \keys_define:nn { enumext / anskey }
2551 {
2552   break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2553   break-col .default:n = true,
2554   break-col .value_forbidden:n = true,
2555   item-join .int_set:N = \l__enumext_store_item_join_int,
2556   item-join .value_required:n = true,
2557   item-star .bool_set:N = \l__enumext_store_item_star_bool,
2558   item-star .default:n = true,

```



```

2559     item-star .value_forbidden:n = true,
2560     item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
2561     item-sym* .value_required:n = true,
2562     item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
2563     item-pos* .value_required:n = true,
2564     unknown .code:n = { \l__enumext_anskey_unknown:n {#1} },
2565 }

```

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

```

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

```

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

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

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

```

2580 \NewDocumentCommand \anskey { o +m }
2581 {
2582     \l__enumext_anskey_safe_outer:
2583     \group_begin:
2584         \bool_if:NT \l__enumext_check_answers_bool
2585         {
2586             \tl_if_novalue:nF {#1}
2587             {
2588                 \keys_set:nn { enumext / anskey } {#1}
2589             }
2590             \tl_if_blank:nTF {#2}
2591             {
2592                 \msg_error:nn { enumext } { anskey-empty-arg }
2593             }
2594             {
2595                 \l__enumext_anskey_safe_inner:
2596                 \l__enumext_store_anskey_code:n {#2}
2597             }
2598         }
2599     \group_end:
2600 }

```

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

12.29.1 Internal functions for the command

`\l__enumext_anskey_safe_outer:` The `\l__enumext_store_anskey_safe_outer:` function will return the appropriate messages when the command is executed outside the environment in which the `save-ans` key was activated.

`\l__enumext_anskey_safe_inner:`

```

2601 \cs_new_protected:Nn \l__enumext_anskey_safe_outer:
2602 {
2603     \bool_if:NF \l__enumext_store_active_bool
2604     {
2605         \msg_error:nnnn { enumext } { anskey-wrong-place } { anskey } { enumext }
2606     }
2607     \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2608     {
2609         \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans }

```

```

2610     }
2611     \int_compare:nNt { \__enumext_keyans_level_h_int } = { 1 }
2612     {
2613         \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans* }
2614     }
2615     \int_compare:nNt { \__enumext_keyans_pic_level_int } = { 1 }
2616     {
2617         \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
2618     }
2619 }

```

The `__enumext_anskey_safe_inner:` function will first check if the command is nested, if preceded by a not numbered `\item` or if it is in *math mode* returning the appropriate messages.

```

2620 \cs_new_protected:Nn \__enumext_anskey_safe_inner:
2621 {
2622     \int_incr:N \__enumext_anskey_level_int
2623     \int_compare:nNt { \__enumext_anskey_level_int } > { 1 }
2624     {
2625         \msg_error:nn { enumext } { anskey-nested }
2626     }
2627     \bool_if:NF \__enumext_item_number_bool
2628     {
2629         \msg_error:nn { enumext } { anskey-unnumber-item }
2630     }
2631     \mode_if_math:T
2632     {
2633         \msg_error:nne { enumext } { anskey-math-mode } { \c_backslash_str anskey }
2634     }
2635 }

```

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

12.30 The environment `anskey*`

Managing *verbatim content* in an environment is quite complicated, I learned that when creating the `scontents` package, so to be able to have support at this point it is best to play a little with the internal code of `scontents` and *hooks*. Some considerations I should have here before implementing this:

- If some package, class or user has defined the environment with the same name somewhere in the document it would be a problem, you would not know what argument has been passed to `store-env`, if you are using the key `print-env` or the `write-out` key, sure, I can detect and modify it within the `enumext` and `enumext*` environments, but it would look strange not to have some keys available when running within these environments.
- A better (perhaps a bit paranoid) option is to define it within the environment in which the `save-ans` key is executed. and have it available only when that key is executed, here I would have absolute control of the *(keys)* and I make sure that `write-out` is not used, then using *hooks after* I undefine it and using *hook before* I check if it has been created by any package, class or user and I return a error, then the user will have to see how to solve the problem.

`__enumext_undefine_anskey_env:`

The function `__enumext_undefine_anskey_env:` will undefine the environment `anskey*` and will be passed to the function `__enumext_execute_after_env:` (§12.31) which is executed after the environment in which the key `save-ans` is active.

```

2636 \cs_new_protected:Nn \__enumext_undefine_anskey_env:
2637 {
2638     \cs_undefine:c { anskey* }
2639     \cs_undefine:c { endanskey* }
2640     \cs_undefine:c { __scontents_anskey*_env_begin: }
2641     \cs_undefine:c { __scontents_anskey*_env_end: }
2642 }

```

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

```

2643 \__enumext_before_env:nn { enumext }
2644 {
2645     \bool_lazy_and:nnT
2646     { \int_compare_p:nNn { \__enumext_level_int } = { 0 } }
2647     { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
2648     {
2649         \cs_if_free:cF { __scontents_anskey*_env_begin: }
2650         {
2651             \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2652         }
2653     }

```

```

2654 }
2655 \__enumext_before_env:nn { enumext* }
2656 {
2657   \bool_lazy_and:nnT
2658   { \int_compare_p:nNn { \__enumext_level_int } = { 0 } }
2659   { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
2660   {
2661     \cs_if_free:cF { __scontents_anskey*_env_begin: }
2662     {
2663       \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2664     }
2665   }
2666 }

```

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.

```

2667 \__enumext_before_env:nn { anskey* }
2668 {
2669   \int_compare:nNnT { \__enumext_keyans_level_int } = { 1 }
2670   {
2671     \msg_error:nnn { enumext } { anskey-env-wrong } { keyans }
2672   }
2673   \int_compare:nNnT { \__enumext_keyans_level_h_int } = { 1 }
2674   {
2675     \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
2676   }
2677   \int_compare:nNnT { \__enumext_keyans_pic_level_int } = { 1 }
2678   {
2679     \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
2680   }
2681   \bool_if:NF \__enumext_item_number_bool
2682   {
2683     \msg_error:nn { enumext } { anskey-unnumber-item }
2684   }
2685   \mode_if_math:T
2686   {
2687     \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
2688   }
2689 }

```

(End of definition for `__enumext_undefine_anskey_env:`)

anskey*

The function `__enumext_anskey_env_make:n` creates the environment `anskey*` (custom version of `scontents` environment) by setting the initial keys `store-env={⟨store name⟩}` and `print-env=false`.

To maintain the *scope* of the environment and that it is only active when the key `save-ans` is active we will pass this function to the function `__enumext_storing_exec:` (§12.25.1) and we will execute it only if the variable `__enumext_anskey_env_bool` is true, with this we prevent it from being executed again when the environment is nested and the key `save-ans` is active, which returns an error for part of the package `scontents`.

```

2690 \cs_new_protected:Npn \__enumext_anskey_env_make:n #1
2691 {
2692   \bool_if:NT \__enumext_anskey_env_bool
2693   {
2694     \newenvsc{anskey*}[store-env=#1,print-env=false]
2695     \__enumext_anskey_env_exec:
2696   }
2697 }
2698 \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`.

```

2699 \cs_new_protected:Nn \__enumext_anskey_env_define_keys:
2700 {
2701   \keys_define:nn { scontents / scontents }
2702   {
2703     break-col .bool_gset:N = \g__enumext_store_columns_break_bool,
2704     break-col .default:n = true,
2705     break-col .value_forbidden:n = true,
2706     item-join .int_gset:N = \g__enumext_store_item_join_int,
2707     item-join .value_required:n = true,

```

```

2708     item-star .bool_gset:N = \g__enumext_store_item_star_bool,
2709     item-star .default:n    = true,
2710     item-star .value_forbidden:n = true,
2711     item-sym* .tl_gset:N    = \g__enumext_store_item_symbol_tl,
2712     item-sym* .value_required:n = true,
2713     item-pos* .dim_gset:N    = \g__enumext_store_item_symbol_sep_dim,
2714     item-pos* .value_required:n = true,
2715     print-env .undefine:,
2716     store-env .undefine:,
2717     write-out .undefine:,
2718     unknown   .code:n       = { \__enumext_anskey_env_unknown:n {##1} },
2719   }
2720 }

```

The *⟨keys⟩* are stored in `\l_keys_key_str` and the value (if any) is passed as an argument to the function `__enumext_anskey_env_unknown:n`.

```

2721 \cs_new_protected:Npn \__enumext_anskey_env_unknown:n #1
2722 {
2723   \exp_args:NV \__enumext_anskey_env_unknown:nn \l_keys_key_str {#1}
2724 }
2725 \cs_new_protected:Npn \__enumext_anskey_env_unknown:nn #1#2
2726 {
2727   \tl_if_blank:nTF {#2}
2728   {
2729     \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
2730   }
2731   {
2732     \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
2733   }
2734 }

```

The function `__enumext_anskey_env_reset_keys:` will leave the keys `break-col`, `item-join`, `item-join`, `item-star`, `item-sym*` and `item-pos*` undefined. We will apply this function using the *hook* function `__enumext_after_env:nn`.

```

2735 \cs_new_protected:Nn \__enumext_anskey_env_reset_keys:
2736 {
2737   \keys_define:nn { scontents / scontents }
2738   {
2739     break-col .undefine:,
2740     item-join .undefine:,
2741     item-star .undefine:,
2742     item-sym* .undefine:,
2743     item-pos* .undefine:,
2744     write-out .code:n    = {
2745                                     \bool_set_false:N \l__scontents_storing_bool
2746                                     \bool_set_true:N  \l__scontents_writing_bool
2747                                     \tl_set:Nn \l__scontents_fname_out_tl {##1}
2748                                 },
2749     write-out .value_required:n = true,
2750     print-env .meta:nn         = { scontents } { print-env = ##1 },
2751     print-env .default:n       = true,
2752     store-env .meta:nn         = { scontents } { store-env = ##1 },
2753     unknown   .code:n          = { \__scontents_parse_environment_keys:n {##1} },
2754   }
2755 }

```

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*.

```

2756 \cs_new_protected:Npn \__enumext_rescan_anskey_env:n #1
2757 {
2758   \group_begin:
2759     \int_set:Nn \tex_newlinechar:D { `^^J }
2760     \__scontents_rescan_tokens:x
2761     {
2762       \endgroup % This assumes \catcode`\=0... Things might go off otherwise.
2763       #1
2764     }
2765 }

```

(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)*.

```

2766 \cs_new_protected:Nn \__enumext_anskey_env_exec:
2767 {
2768   \__enumext_before_env:nn { anskey* }
2769   {
2770     \__enumext_anskey_env_define_keys:
2771   }

```

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.

```

2772   \hook_if_empty:nF {env/anskey*/after}
2773   {
2774     \hook_gremove_code:nn {env/anskey*/after} { * }
2775   }
2776   \__enumext_after_env:nn { anskey* }
2777   {
2778     \__enumext_anskey_env_save_keys:
2779     \tl_clear:N \l__enumext_store_anskey_env_tl
2780     \tl_clear:N \l__enumext_store_anskey_opt_tl
2781     \bool_if:NT \l__enumext_check_answers_bool
2782     {
2783       \tl_gset:Ne \l__enumext_store_anskey_env_tl
2784       {
2785         \seq_item:ce { g__scontents_name_ \l__enumext_store_name_tl _seq } { -1 }
2786       }
2787       \regex_match:nVTF
2788       { ^\s* \z | ^\s* \u{c__scontents_hidden_space_str} \z }
2789       \l__enumext_store_anskey_env_tl
2790       {
2791         \msg_error:nn { enumext } { anskey-empty-arg }
2792       }
2793       {
2794         \__enumext_anskey_env_store:
2795       }
2796     }
2797     \__enumext_anskey_env_clean_vars:
2798     \__enumext_anskey_env_reset_keys:
2799   }
2800 }

```

• The use of `\hook_gremove_code:nn` is necessary here, otherwise the *{⟨code⟩}* passed to `__enumext_after_env:nn{anskey*}` will be accumulated for each execution. The last function `__enumext_anskey_env_reset_keys:` is necessary so as not to hinder any *scontents* environment running within *enumext* or *enumext**.

(End of definition for `__enumext_anskey_env_exec:`.)

`__enumext_anskey_env_save_keys:` The function `__enumext_anskey_env_save_keys:` processing the *[⟨key = val⟩]* passed to the environment and save this in the variable `\l__enumext_store_anskey_opt_tl`. If the *break-col* key is present and the environment is running under *enumext* (not in *enumext**) we will add the key *break-col*.

`__enumext_anskey_env_store:`

`__enumext_anskey_env_clean_vars:`

```

2801 \cs_new_protected:Nn \__enumext_anskey_env_save_keys:
2802 {
2803   \bool_lazy_and:nnT
2804   { \bool_if_p:N \g__enumext_store_columns_break_bool }
2805   { \bool_not_p:n { \l__enumext_starred_bool } }
2806   {
2807     \tl_put_left:Ne \l__enumext_store_anskey_opt_tl { ,break-col, }
2808   }

```

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

```

2809   \bool_lazy_and:nnT
2810   { \bool_not_p:n { \l__enumext_starred_bool } }
2811   { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }
2812   {
2813     \tl_put_left::Ne \l__enumext_store_anskey_opt_tl
2814     {
2815       ,item-join = \exp_not:V \g__enumext_store_item_join_int,
2816     }
2817   }

```

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

```

2818   \bool_if:NT \g__enumext_store_item_star_bool
2819   {
2820     \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2821     {
2822       ,item-star,
2823     }
2824     \tl_if_empty:NF \g__enumext_store_item_symbol_tl
2825     {
2826       \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2827       {
2828         ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,
2829       }
2830     }
2831     \dim_compare:nT
2832     {
2833       \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
2834     }
2835     {
2836       \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2837       {
2838         ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
2839       }
2840     }
2841   }
2842 }

```

The function `__enumext_anskey_env_store:` will be responsible for storing the content of the environment using the functions `__enumext_store_anskey_code:n` and `__enumext_rescan_anskey_env:n`.

```

2843 \cs_new_protected:Nn \__enumext_anskey_env_store:
2844 {
2845   \group_begin:
2846   \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
2847   {
2848     \exp_args:Ne
2849     \__enumext_store_anskey_code:n
2850     {
2851       \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2852     }
2853   }
2854   {
2855     \keys_set_known:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
2856     \exp_args:Ne
2857     \__enumext_store_anskey_code:n
2858     {
2859       \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2860     }
2861   }
2862   \group_end:
2863 }

```

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

```

2864 \cs_new_protected:Nn \__enumext_anskey_env_clean_vars:
2865 {
2866   \bool_gset_false:N \g__enumext_store_columns_break_bool
2867   \int_gzero:N       \g__enumext_store_item_join_int
2868   \bool_gset_false:N \g__enumext_store_item_star_bool
2869   \tl_gclear:N       \g__enumext_store_item_symbol_tl
2870   \dim_gzero:N       \g__enumext_store_item_symbol_sep_dim
2871 }

```

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

12.31 Executing `anskey*`, `check-ans` and write `.log`

`__enumext_execute_after_env:`

The `__enumext_execute_after_env:` function will first return the appropriate message for the end of the environment in which the `save-ans` key is being executed, then call the `__enumext_item_answer_diff:` function and then will write the values of the global variables used to the `.log` file. If the key `check-ans` is active it will execute the function `__enumext_check_ans_show:` and show the result in the terminal,

otherwise it will execute the function `__enumext_check_ans_log:` and write the results in the `.log` file, undefine the environment `anskey*` (§12.30) through the function `__enumext_undefine_anskey_env:` and finally we execute the function `__enumext_reset_global_vars:` returning the used variables to their original state.

```

2872 \cs_new_protected:Nn \__enumext_execute_after_env:
2873 {
2874   \int_compare:nNnT { \__enumext_level_int } = { 0 }
2875   {
2876     \tl_if_empty:NF \g__enumext_store_name_tl
2877     {
2878       \__enumext_stop_save_ans_msg:
2879       \__enumext_item_answer_diff:
2880       \__enumext_log_global_vars:
2881       \__enumext_log_answer_vars:
2882       \bool_if:NTF \g__enumext_check_ans_key_bool
2883       {
2884         \__enumext_check_ans_show:
2885       }
2886       { \__enumext_check_ans_log: }
2887       \__enumext_undefine_anskey_env:
2888     }
2889     \__enumext_reset_global_vars:
2890   }
2891 }

```

(End of definition for `__enumext_execute_after_env:`.)

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

12.32 Common functions for `keyans`, `keyans*` and `keyanspic`

12.32.1 Storing content in prop list

`__enumext_keyans_addto_prop:n`

The function `__enumext_keyans_addto_prop:n` will pass the contents of the current `<label>` `\l__enumext_label_v_tl` for the `keyans` environment and the current `<label>` `\l__enumext_label_vi_tl` for the `keyanspic` environment when using `\item*` and `\anspic*`, followed by the *contents* of the optional argument of both commands to the `\l__enumext_store_current_label_tl` variable, which will be passed to the *<prop list>* defined by the `save-ans` key using the `__enumext_store_addto_prop:V`.

```

2892 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2893 {
2894   \tl_clear:N \l__enumext_store_current_label_tl
2895   \int_compare:nNnTF { \__enumext_keyans_pic_level_int } = { 1 }
2896   {
2897     \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
2898   }
2899   {
2900     \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
2901   }
2902   \tl_if_novalue:NF { #1 }
2903   {
2904     % Set save-sep
2905     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2906     {
2907       \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
2908     }
2909     \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2910   }
2911   \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
2912 }

```

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

12.32.2 The `save-ref` key for `keyans`, `keyans*` and `keyanspic`

The “*internal label and ref*” system for the `keyans`, `keyans*` and `keyanspic` environments has slight differences with the one implemented for the `\anskey` command, basically because in this environments we are interested in the current `<label>`. The mechanism defined here will allow to execute `\ref{<store name : position>}` and will return `1.` (A).

`__enumext_keyans_store_ref:`
`__enumext_keyans_store_ref_aux_i:`
`__enumext_keyans_store_ref_aux_ii:`

The function `__enumext_keyans_store_ref:` handles the internal “*label and ref*” system used by the `save-ref` key for `\item*` and `\anspic*` commands. First we will create copies of the current `<labels>` and remove the dots “.” from them, we do not want to get double dots in our references.


```

2913 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2914 {
2915   \bool_if:NT \l__enumext_store_ref_key_bool
2916   {
2917     \cs_set_protected:Npn \__enumext_tmp:n #1
2918     {
2919       \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2920       \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2921       \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2922       \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2923     }
2924     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2925     \__enumext_keyans_store_ref_aux_i:
2926   }
2927 }

```

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

```

2928 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2929 {
2930   \bool_if:NT \g__enumext_starred_bool
2931   {
2932     \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2933   }
2934   \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
2935   {
2936     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2937     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
2938   }
2939   \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
2940   {
2941     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2942     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2943   }
2944   \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
2945   {
2946     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2947     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2948   }
2949   \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2950   {
2951     \l__enumext_store_name_tl \c_colon_str
2952     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2953   }
2954   \__enumext_keyans_store_ref_aux_ii:
2955 }

```

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.

```

2956 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2957 {
2958   \tl_put_right:Ne \l__enumext_write_aux_file_tl
2959   {
2960     \__enumext_newlabel:nn
2961     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2962     { \l__enumext_newlabel_arg_two_tl }
2963   }
2964   \l__enumext_write_aux_file_tl
2965 }

```

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

12.32.3 Storing content in sequence

```

\__enumext_keyans_addto_seq:n
\__enumext_keyans_addto_seq_link:

```

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

```

2966 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1

```

```

2967 {
2968   \tl_clear:N \l__enumext_store_current_label_tl
2969   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2970   {
2971     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
2972   }
2973   {
2974     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
2975   }
2976   \tl_if_novalue:nF { #1 }
2977   {
2978     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2979     {
2980       \tl_put_right:Ne \l__enumext_store_current_label_tl
2981       {
2982         \l__enumext_store_keyans_item_opt_sep_tl
2983       }
2984     }
2985     \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2986   }
2987   \__enumext_keyans_addto_seq_link:
2988 }

```

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.

```

2989 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2990 {
2991   \bool_lazy_and:nnT
2992   { \bool_if_p:N \l__enumext_store_ref_key_bool }
2993   { \bool_if_p:N \l__enumext_hyperref_bool }
2994   {
2995     \tl_put_right:Ne \l__enumext_store_current_label_tl
2996     {
2997       \hfill \exp_not:N \hyperlink
2998       {
2999         \exp_not:V \l__enumext_newlabel_arg_one_tl
3000       }
3001       { \exp_not:V \l__enumext_mark_ref_sym_tl }
3002     }
3003   }
3004   \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
3005   \bool_if:NT \l__enumext_check_answers_bool
3006   {
3007     \int_gincr:N \g__enumext_item_anskey_int
3008   }
3009 }

```

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

12.32.4 The show-ans and show-pos keys for keyans and keyanspic

The code is very similar to the `\anskey` code, but, if I change the order of the operations the counter off `\label` are incorrect.

```

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

```

Common function to show *starred commands* `\item*` and `\position` of stored content in `\prop list` for `keyans` and `keyanspic`. Need add `1` to `\g__enumext_<store name>_prop` for `show-pos` key.

```

3010 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
3011 {
3012   \tl_if_novalue:nF { #1 }
3013   {
3014     \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
3015   }
3016   \bool_if:NT \l__enumext_show_answer_bool
3017   {
3018     \__enumext_keyans_show_ans:
3019   }
3020   \bool_if:NT \l__enumext_show_position_bool
3021   {

```

```

3022     \__enumext_keyans_show_pos:
3023   }
3024 }
3025 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
3026 {
3027   \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
3028   {
3029     \bool_lazy_or:nnT
3030     { \bool_if_p:N \l__enumext_show_answer_bool }
3031     { \bool_if_p:N \l__enumext_show_position_bool }
3032     {
3033       \__enumext_keyans_wrapper_opt:n { \l__enumext_store_current_opt_arg_tl } \c_space_tl
3034     }
3035   }
3036 }
3037 \cs_new_protected:Nn \__enumext_keyans_show_ans:
3038 {
3039   \bool_if:NT \l__enumext_starred_bool
3040   {
3041     \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
3042     \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
3043   }
3044   \tl_put_left:Nn \l__enumext_label_v_tl
3045   {
3046     \__enumext_print_keyans_box:NN
3047     \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3048   }
3049 }
3050 \cs_new_protected:Nn \__enumext_keyans_show_pos:
3051 {
3052   \bool_if:NT \l__enumext_starred_bool
3053   {
3054     \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
3055     \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
3056   }
3057   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
3058   {
3059     \tl_set:Ne \l__enumext_mark_answer_sym_tl
3060     {
3061       \group_begin:
3062       \exp_not:N \normalfont
3063       \exp_not:N \footnotesize [ \int_eval:n
3064         {
3065           \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3066         }
3067       ]
3068       \group_end:
3069     }
3070   }
3071   {
3072     \tl_set:Ne \l__enumext_mark_answer_sym_tl
3073     {
3074       \group_begin:
3075       \exp_not:N \normalfont
3076       \exp_not:N \footnotesize [ \int_eval:n
3077         {
3078           \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
3079         }
3080       ]
3081       \group_end:
3082     }
3083   }
3084   \tl_put_left:Nn \l__enumext_label_v_tl
3085   {
3086     \__enumext_print_keyans_box:NN
3087     \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3088   }
3089 }

```

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

12.33 Redefining \item and \makeLabel in enumext

Redefining the `\item` command is not as simple as I thought. This command works in conjunction with the `\makeLabel` command so I have to redefine both of them, in addition to this, we will have to use a couple of *global* variables to pass the values from one command to the other.

The `\item` and `\item[⟨custom⟩]` commands work in the usual way on `enumext` and we will add `\item*`, `\item*[⟨symbol⟩]` and `\item*[⟨symbol⟩][⟨offset⟩]`.

`__enumext_default_item:n`

First we will see if the optional argument is present, if it is NOT present we will check the state of the variable `\l__enumext_check_answers_bool` set by the key `no-store`, set the boolean variable `\l__enumext_wrap_label_X_bool` to “true” for the key `wrap-label` and execute `__enumext_item_std:w` and the key `itemindent`, otherwise we will check the state of the boolean variable `\l__enumext_wrap_label_opt_X_bool` set by the key `wrap-label*` and execute `__enumext_item_std:w` with the optional argument and the key `itemindent`.

```

3090 \cs_new_protected:Npn \__enumext_default_item:n #1
3091 {
3092   \tl_if_novalue:nTF {#1}
3093   {
3094     \bool_if:NT \l__enumext_check_answers_bool
3095     {
3096       \int_gincr:N \g__enumext_item_number_int
3097       \bool_set_true:N \l__enumext_item_number_bool
3098     }
3099     \bool_set_true:c { \l__enumext_wrap_label_ \__enumext_level: _bool }
3100     \__enumext_item_std:w \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
3101   }
3102   {
3103     \bool_set_eq:cc
3104     { \l__enumext_wrap_label_ \__enumext_level: _bool }
3105     { \l__enumext_wrap_label_opt_ \__enumext_level: _bool }
3106     \__enumext_item_std:w [#1] \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
3107   }
3108 }

```

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

`__enumext_starred_item:nn`

`__enumext_item_star_exec:`

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

`#1: \l__enumext_item_symbol_X_tl`

`#2: \l__enumext_item_symbol_sep_X_dim`

First we will make a copy of `\l__enumext_item_symbol_X_tl` which is set by the key `item-sym*` or passed as “first” optional argument in the global variable `\g__enumext_item_symbol_aux_tl`, followed by setting the variable `\l__enumext_item_symbol_sep_X_dim` set by the key `item-pos*` or by the “second” optional argument, then we will see the state of the variable `\l__enumext_check_answers_bool` set by the key `no-store`, set the boolean variable `\l__enumext_wrap_label_X_bool` to “true” for the key `wrap-label` and execute `__enumext_item_std:w` and the key `itemindent`.

```

3109 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
3110 {
3111   \tl_if_novalue:nTF {#1}
3112   {
3113     \tl_gset_eq:Nc
3114     \g__enumext_item_symbol_aux_tl { \l__enumext_item_symbol_ \__enumext_level: _tl }
3115   }
3116   {
3117     \tl_gset:Nn \g__enumext_item_symbol_aux_tl {#1}
3118   }
3119   \tl_if_novalue:nTF {#2}
3120   {
3121     \dim_set_eq:cc
3122     { \l__enumext_item_symbol_sep_ \__enumext_level: _dim }
3123     { \l__enumext_labelsep_ \__enumext_level: _dim }
3124   }
3125   {
3126     \dim_set:cn { \l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
3127   }
3128   \bool_if:NT \l__enumext_check_answers_bool
3129   {
3130     \int_gincr:N \g__enumext_item_number_int
3131     \bool_set_true:N \l__enumext_item_number_bool

```

```

3132     }
3133     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
3134     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3135 }

```

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

```

3136 \cs_new_protected:Nn \__enumext_item_star_exec:
3137 {
3138   \tl_if_empty:cF { l__enumext_item_symbol_ \__enumext_level: _tl }
3139   {
3140     \mode_leave_vertical:
3141     \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3142     \hbox_overlap_left:n { \g__enumext_item_symbol_aux_tl }
3143     \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3144   }
3145 }

```

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

```

\__enumext_redefine_item:
\__enumext_make_label

```

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

```

3146 \cs_new_protected:Nn \__enumext_redefine_item:
3147 {
3148   \RenewDocumentCommand \item { s o o }
3149   {
3150     \bool_if:nTF {##1}
3151     {
3152       \__enumext_starred_item:nn {##2} {##3}
3153     }
3154     { \__enumext_default_item:n {##2} }
3155   }
3156 }

```

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

```

3157 \cs_new_protected:Nn \__enumext_make_label:
3158 {
3159   \RenewDocumentCommand \make_label { m }
3160   {
3161     \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
3162     \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3163     \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3164     {
3165       \__enumext_item_star_exec:
3166       \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3167     }
3168     { ##1 }
3169     \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
3170     \tl_gclear:N \g__enumext_item_symbol_aux_tl
3171   }
3172 }

```

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

🔗 This functions are passed to `__enumext_list_arg_two_X:` used in the definition of the `enumext` environment (§12.38).

12.34 Setting `item-sym*` and `item-pos*` keys

In order to have a cleaner implementation of `\item*` for the `enumext` and `enumext*` environments it is best to define a couple of keys that allow us to control and set by default the `<symbol>` and its `<offset>`.

`item-sym*`

Define and set `item-sym*` and `item-pos*` keys for `enumext` and `enumext*`.

`item-pos*`

```

3173 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
3174 {
3175   \keys_define:nn { enumext / #1 }
3176   {
3177     item-sym* .tl_set:c = { l__enumext_item_symbol_#2_tl },
3178     item-sym* .value_required:n = true,
3179     item-sym* .initial:n = { $\star$ },
3180     item-pos* .dim_set:c = { l__enumext_item_symbol_sep_#2_dim },
3181     item-pos* .value_required:n = true,
3182   }

```

```

3183     }
3184 \clist_map_inline:nn
3185 {
3186     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
3187 }
3188 { \__enumext_tmp:nn #1 }

```

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

12.35 Handling unknown keys

At this point in the code I already know that I will not add more `<keys>` and since I have already been quite *paranoid and restrictive* with the definitions of environments and commands, the only thing left to do is do it with the `<keys>` (you have to be consistent in life).

12.35.1 Handling unknown keys for `keyans` and `keyans*`

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

```

unknown
\__enumext_keyans_unknown_keys:n
\__enumext_keyans_unknown_keys:nn
3189 \cs_set_protected:Npn \__enumext_tmp:n #1
3190 {
3191     \keys_define:nn { enumext / #1 }
3192     {
3193         unknown .code:n = { \__enumext_keyans_unknown_keys:n {##1} }
3194     }
3195 }
3196 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }

```

Internal functions for handling `unknown` key.

```

3197 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:n #1
3198 {
3199     \exp_args:NV \__enumext_keyans_unknown_keys:nn \l_keys_key_str {#1}
3200 }
3201 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:nn #1#2
3202 {
3203     \tl_if_blank:nTF {#2}
3204     {
3205         \msg_error:nnn { enumext } { keyans-unknown-key } {#1}
3206     }
3207     {
3208         \msg_error:nnnn { enumext } { keyans-unknown-key-value } {#1} {#2}
3209     }
3210 }

```

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

12.35.2 Handling unknown keys for `enumext*`

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

```

unknown
\__enumext_starred_unknown_keys:n
\__enumext_starred_unknown_keys:nn
3211 \keys_define:nn { enumext / enumext* }
3212 {
3213     unknown .code:n = { \__enumext_starred_unknown_keys:n {#1} }
3214 }

```

Internal functions for handling `unknown` key.

```

3215 \cs_new_protected:Npn \__enumext_starred_unknown_keys:n #1
3216 {
3217     \exp_args:NV \__enumext_starred_unknown_keys:nn \l_keys_key_str {#1}
3218 }
3219 \cs_new_protected:Npn \__enumext_starred_unknown_keys:nn #1#2
3220 {
3221     \tl_if_blank:nTF {#2}
3222     {
3223         \msg_error:nnn { enumext } { starred-unknown-key } {#1}
3224     }
3225     {
3226         \msg_error:nnnn { enumext } { starred-unknown-key-value } {#1} {#2}
3227     }
3228 }

```

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

12.35.3 Handling unknown keys for enumext

unknown

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

```

3229 \cs_set_protected:Npn \__enumext_tmp:n #1
3230 {
3231   \keys_define:nn { enumext / #1 }
3232   {
3233     unknown .code:n = { \__enumext_standar_unknown_keys:n {##1} }
3234   }
3235 }
3236 \clist_map_inline:nn { level-1,level-2,level-3,level-4 } { \__enumext_tmp:n {#1} }
```

Internal functions for handling `unknown` key.

```

3237 \cs_new_protected:Npn \__enumext_standar_unknown_keys:n #1
3238 {
3239   \exp_args:NV \__enumext_standar_unknown_keys:nn \l_keys_key_str {#1}
3240 }
3241 \cs_new_protected:Npn \__enumext_standar_unknown_keys:nn #1#2
3242 {
3243   \tl_if_blank:nTF {#2}
3244   {
3245     \msg_error:nnn { enumext } { standar-unknown-key } {#1}
3246   }
3247   {
3248     \msg_error:nnnn { enumext } { standar-unknown-key-value } {#1} {#2}
3249   }
3250 }
```

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

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

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

`__enumext_keyans_default_item:n`

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

```

3251 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
3252 {
3253   \tl_if_novalue:nTF { #1 }
3254   {
3255     \bool_set_true:N \__enumext_wrap_label_v_bool
3256     \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl
3257   }
3258   {
3259     \bool_set_eq:NN \__enumext_wrap_label_v_bool \__enumext_wrap_label_opt_v_bool
3260     \__enumext_item_std:w [#1] \tl_use:N \__enumext_fake_item_indent_v_tl
3261   }
3262 }
```

(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⟩`.

```

3263 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
3264 {
3265   \tl_set_eq:NN \__enumext_store_current_label_tmp_tl \__enumext_label_v_tl
3266   \__enumext_keyans_show_left:n { #1 }
3267   \bool_set_true:N \__enumext_wrap_label_v_bool
3268   \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl \__enumext_keyans_show_item
```

Recover the original value of the current `⟨label⟩` and *store* it first in the `⟨prop list⟩` (including the optional argument), run the internal “*label and ref*” system if the `save-ref` key is active and finally *store* it in the `⟨sequence⟩`.

```

3269   \tl_set_eq:NN \__enumext_label_v_tl \__enumext_store_current_label_tmp_tl
3270   \__enumext_keyans_addto_prop:n { #1 }
3271   \__enumext_keyans_store_ref:
3272   \__enumext_keyans_addto_seq:n { #1 }
3273   \int_gincr:N \g__enumext_check_starred_cmd_int
3274 }
```


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

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

```

3275 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
3276 {
3277   \RenewDocumentCommand \item { s o }
3278   {
3279     \bool_if:nTF {##1}
3280     {
3281       \peek_remove_spaces:n
3282       {
3283         \__enumext_keyans_starred_item:n {##2}
3284       }
3285     }
3286     {
3287       \__enumext_keyans_default_item:n {##2}
3288     }
3289   }
3290 }

```

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

```

3291 \cs_new_protected:Nn \__enumext_keyans_make_label:
3292 {
3293   \RenewDocumentCommand \makeLabel { m }
3294   {
3295     \tl_use:N \l__enumext_label_fill_left_v_tl
3296     \tl_use:N \l__enumext_label_font_style_v_tl
3297     \bool_if:nTF \l__enumext_wrap_label_v_bool
3298     {
3299       \__enumext_wrapper_label_v:n { ##1 }
3300     }
3301     { ##1 }
3302     \tl_use:N \l__enumext_label_fill_right_v_tl
3303   }
3304 }

```

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

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

12.37 Second argument of the lists

At this point of the code we have already programmed most the necessary tools to create a custom `list` environment, remember that the function `__enumext_start_list:nn` takes two arguments, the first one we have ready, the second one we will define for all the levels of the environment `enumext` and the environment `keyans`.

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

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

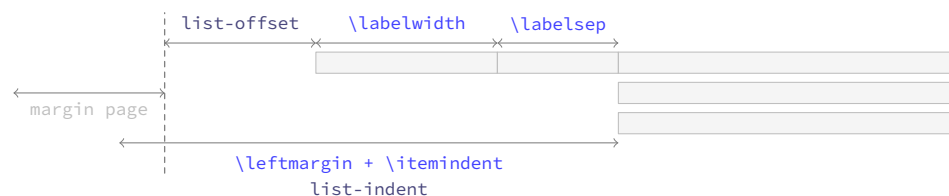


Figure 9: Representation of standard horizontal lengths in `list` environment.

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

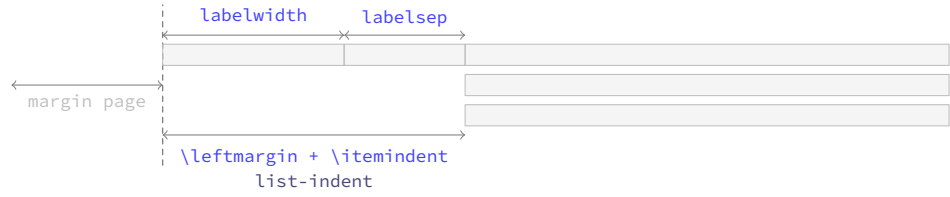
Where the default values will look like in the figure 11.

```

\__enumext_calc_hspace:NNNNNNN
\__enumext_calc_hspace:ccccccc

```

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

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

```

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

```

And returns the “adjusted” values of `\leftmargin` and `\itemindent`.

This function is passed to `__enumext_list_arg_two_X`: which is used in the definition of the `enumext` and `keyans` environments (§12.37.2).

```

3305 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNN #1 #2 #3 #4 #5 #6 #7
3306 {
3307   \dim_compare:nNnT { #1 } < { \c_zero_dim }
3308   {
3309     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
3310     \dim_set:Nn #1 { \dim_abs:n { #1 } }
3311   }
3312   \dim_compare:nNnT { #2 } < { \c_zero_dim }
3313   {
3314     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
3315     \dim_set:Nn #2 { \dim_abs:n { #2 } }
3316   }

```

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

```

3317   \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }

```

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

```

3318   \dim_compare:nNnTF { #4 } < { \c_zero_dim }
3319   {
3320     \dim_set:Nn #6 { #1 + #2 - #4 }
3321     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3322   }
3323   {
3324     \dim_compare:nNnT { #4 } = { #1 + #2 }
3325     { \dim_set:Nn #6 { \c_zero_dim } }
3326     \dim_compare:nNnT { #4 } < { #1 + #2 }
3327     { \dim_set:Nn #6 { #1 + #2 - #4 } }
3328     \dim_compare:nNnT { #4 } > { #1 + #2 }
3329     {
3330       \dim_set:Nn #6 { -#1 - #2 + #4 }
3331       \dim_set:Nn #6 { #6*-1 }
3332     }
3333     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3334   }
3335 }
3336 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNN { ccccccc }

```

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

12.37.2 Setting second argument of the lists

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

```

3337 \cs_set_protected:Npn \__enumext_tmp:n #1
3338 {
3339   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3340   {
3341     \__enumext_calc_hspace:ccccc
3342     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3343     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3344     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3345     { \__enumext_leftmargin_tmp_#1_bool }
3346     \clist_map_inline:nn
3347       { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3348       { \dim_set_eq:cc {###1} { \__enumext_###1_#1_dim } }
3349     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3350       { \skip_set_eq:cc {###1} { \__enumext_###1_#1_skip } }
3351     \usecounter { enumX#1 }
3352     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
3353     \str_if_eq:nnTF {#1} { v }
3354     {
3355       \__enumext_keyans_redefine_item:
3356       \__enumext_keyans_make_label:
3357       \__enumext_keyans_ref:
3358       \__enumext_keyans_fake_item:
3359       \bool_if:cT { \__enumext_show_length_#1_bool }
3360       {
3361         \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3362       }
3363     }
3364     {
3365       \__enumext_redefine_item:
3366       \__enumext_make_label:
3367       \__enumext_standar_ref:
3368       \__enumext_fake_item:
3369       \bool_if:cT { \__enumext_show_length_#1_bool }
3370       {
3371         \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \__enumext_level_int }
3372       }
3373     }
3374   }
3375 }
3376 \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 `\listparindent` and `parsep` to set the value of `\parskip` locally.

```

3377 \cs_set_protected:Npn \__enumext_tmp:n #1
3378 {
3379   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3380   {
3381     \bool_set_true:c { \__enumext_leftmargin_tmp_#1_bool }
3382     \dim_zero:c { \__enumext_leftmargin_tmp_#1_dim }
3383     \__enumext_calc_hspace:ccccc
3384     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3385     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3386     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3387     { \__enumext_leftmargin_tmp_#1_bool }
3388     \clist_map_inline:nn
3389       { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3390       { \dim_set_eq:cc {###1} { \__enumext_###1_#1_dim } }
3391     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3392       { \skip_set_eq:cc {###1} { \__enumext_###1_#1_skip } }
3393     \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
3394     \skip_zero:N \partopsep
3395     \usecounter { enumX#1 }
3396     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }

```

```

3397     \__enumext_starred_ref:
3398     \str_if_eq:nnTF {#1} { vii }
3399     {
3400         \__enumext_fake_item_vii:
3401         \bool_if:cT { \__enumext_show_length_vii_bool }
3402         { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3403     }
3404     {
3405         \__enumext_fake_item_viii:
3406         \bool_if:cT { \__enumext_show_length_#1_bool }
3407         { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3408     }
3409 }
3410 }
3411 \clist_map_inline:nn { vii, viii } { \__enumext_tmp:n {#1} }

```

(End of definition for __enumext_list_arg_two_vii: and __enumext_list_arg_two_viii:.)

12.38 The environment enumext

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

```

3412 \NewDocumentEnvironment{enumext}{ 0{ } }
3413 {
3414     \__enumext_safe_exec:
3415     \__enumext_parse_keys:n {#1}
3416     \__enumext_before_list:
3417     \__enumext_start_store_level:
3418     \__enumext_start_list:nn
3419     { \tl_use:c { \__enumext_label_ \__enumext_level: _tl } }
3420     {
3421         \use:c { __enumext_list_arg_two_ \__enumext_level: : }
3422         \__enumext_before_keys_exec:
3423     }
3424     \__enumext_set_item_width:
3425     \__enumext_after_args_exec:
3426 }
3427 {
3428     \__enumext_stop_list:
3429     \__enumext_stop_store_level:
3430     \__enumext_after_list:
3431 }

```

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

`__enumext_set_item_width:` The function `__enumext_set_item_width:` will set the value of `\itemwidth` taking into account the value established by the `list-offset` key for each level of the environment.

```

3432 \cs_new_protected:Nn \__enumext_set_item_width:
3433 {
3434     \dim_set:Nn \itemwidth
3435     {
3436         \linewidth
3437     }
3438     \dim_compare:nT
3439     {
3440         \dim_use:c { \__enumext_listoffset_ \__enumext_level: _dim } != \c_zero_dim
3441     }
3442     {
3443         \dim_sub:Nn \itemwidth
3444         {
3445             \dim_use:c { \__enumext_listoffset_ \__enumext_level: _dim }
3446         }
3447     }
3448 }

```

(End of definition for `__enumext_set_item_width:`.)

`__enumext_safe_exec:` The `__enumext_safe_exec:` function first call the function `__enumext_internal_mini_page:` to create the environment `__enumext_mini_page`, then the function `__enumext_is_not_nested:` which sets `\g__enumext_standar_bool` to “true” if we are not nested within `enumext*`, we will increment `__enumext_level_int` to restrict nesting of the environment, set `__enumext_standar_bool` to “true” and

finally call the function `__enumext_is_on_first_level:` which sets `\l__enumext_standar_first_bool` to “true” only if the environment is not nested and we are at the “first level”.

```

3449 \cs_new_protected:Nn \__enumext_safe_exec:
3450 {
3451   \__enumext_internal_mini_page:
3452   \__enumext_is_not_nested:
3453   \int_incr:N \__enumext_level_int
3454   \int_compare:nNnT { \__enumext_level_int } > { 4 }
3455     { \msg_fatal:nn { enumext } { list-too-deep } }
3456   \bool_set_true:N \__enumext_standar_bool
3457   \bool_set_false:N \__enumext_starred_bool
3458   \__enumext_is_on_first_level:
3459 }

```

(End of definition for `__enumext_safe_exec:`)

`__enumext_parse_keys:n`

The `__enumext_parse_store_keys:n` function first we will clear the variable `\l__enumext_series_str` used by the key `series` and then we check if we are at the “first level”, if so we process the `(keys)` and then execute the function `__enumext_parse_series:n` used by the key `series` and call the function `__enumext_nested_base_line_fix:` used by the key `base-fix`, otherwise we will pass the `(keys)` to the inner levels of the environment then we execute the function `__enumext_store_active_keys:n` and reprocess the `(keys)` to pass them to the storage `(sequence)` if the key `save-key` is not active.

```

3460 \cs_new_protected:Npn \__enumext_parse_keys:n #1
3461 {
3462   \tl_if_novalue:nF {#1}
3463   {
3464     \str_clear:N \l__enumext_series_str
3465     \int_compare:nNnTF { \__enumext_level_int } = { 1 }
3466     {
3467       \keys_set:nn { enumext / level-1 } {#1}
3468       \__enumext_parse_series:n {#1}
3469       \__enumext_nested_base_line_fix:
3470     }
3471     {
3472       \exp_args:Ne \keys_set:nn
3473         { enumext / level-\int_use:N \__enumext_level_int } {#1}
3474     }
3475     \__enumext_store_active_keys:n {#1}
3476   }
3477 }

```

(End of definition for `__enumext_parse_keys:n`)

`__enumext_start_store_level:`

The `__enumext_start_store_level:` and `__enumext_stop_store_level:` functions activate the level saving mechanism for storage in `(sequence)` for the command `\anskey` and the environment `anskey*`.

```

3478 \cs_new_protected:Nn \__enumext_start_store_level:
3479 {
3480   \bool_lazy_all:nT
3481   {
3482     { \bool_if_p:N \__enumext_store_active_bool }
3483     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3484     { \bool_if_p:N \g__enumext_standar_bool }
3485   }
3486   {
3487     \int_compare:nNnT { \__enumext_level_int } > { 1 }
3488     {
3489       \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3490       \__enumext_store_level_open:
3491     }
3492   }

```

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

```

3493   \bool_lazy_all:nT
3494   {
3495     { \bool_if_p:N \__enumext_store_active_bool }
3496     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3497     { \int_compare_p:nNn { \__enumext_level_h_int } = { 1 } }
3498   }
3499   {
3500     \int_compare:nNnT { \__enumext_level_int } > { 0 }
3501     {

```

```

3502         \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3503         \__enumext_store_level_open:
3504     }
3505 }
3506 }

```

Close the stored structure.

```

3507 \cs_new_protected:Nn \__enumext_stop_store_level:
3508 {
3509     \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
3510     {
3511         \__enumext_store_level_close:
3512     }
3513 }

```

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

`__enumext_before_list:` The function `__enumext_before_list:` first calls the function `__enumext_vspace_above:` used by the keys `above` and `above*`, then calls the function `__enumext_before_args_exec:` used by the key `before*` and finally execute the function `__enumext_check_ans_active:` for the check answer mechanism.

```

3514 \cs_new_protected:Nn \__enumext_before_list:
3515 {
3516     \__enumext_vspace_above:
3517     \__enumext_before_args_exec:
3518     \__enumext_check_ans_active:

```

When the `mini-env` key is active it will set the value of the `\l__enumext_minipage_right_X_dim` to be the *width* of the `__enumext_mini_page` environment on the “right side”, using this value together with the value of the `\l__enumext_minipage_hsep_X_dim` set by the `mini-sep` key, the value of `\l__enumext_minipage_left_X_dim` will be set, which will be the *width* of `__enumext_mini_page` environment on the “left side”, always having a current `\linewidth` as *maximum width* between them.

```

3519     \dim_compare:nNt
3520     { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3521     {
3522         \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
3523         {
3524             \linewidth
3525             - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
3526             - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
3527         }

```

The boolean variable `\l__enumext_minipage_active_X_bool` will be activated and the integer variable `\g__enumext_minipage_stat_int` used by the `\miniright` command will be incremented, then the function `__enumext_minipage_add_space:` is called and the `__enumext_mini_page` environment on the “left side” will be initialized followed by the “vertical spacing” applied to preserve the “baseline” between the *left* and *right* side environments. After these actions, the function `__enumext_multicols_start:` is called to handle the `multicols` environment.

```

3528         \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
3529         \int_gincr:N \g__enumext_minipage_stat_int
3530         \__enumext_minipage_add_space:
3531         \__enumext_mini_page{ \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
3532     }
3533     \__enumext_multicols_start:
3534 }

```

(End of definition for __enumext_before_list:.)

`__enumext_multicols_start:` The function `__enumext_multicols_start:` will start the `multicols` environment according to the value passed by the `columns` key, then set the default value for `\columnsep` when `columns-sep=opt` and set the value of `\multicolsep` equal to zero and leave `\columnseprule` equal to zero for inner levels.

```

3535 \cs_new_protected:Nn \__enumext_multicols_start:
3536 {
3537     \int_compare:nNt
3538     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3539     {
3540         \dim_compare:nNt
3541         { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3542         {
3543             \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
3544             {
3545                 ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }

```

```

3546         + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
3547     ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
3548     - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3549 }
3550 }
3551 \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
3552 \int_compare:nNt { \l__enumext_level_int } > { 1 }
3553 {
3554     \dim_zero:N \columnseprule
3555 }

```

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.

```

3556 \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3557 {
3558     \skip_zero:N \multicolsep
3559     \__enumext_multi_addvspace:
3560 }
3561 \raggedcolumns
3562 \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
3563 }
3564 }

```

(End of definition for `__enumext_multicols_start:`)

`__enumext_multicols_stop:` The function `__enumext_multicols_stop:` will stop the `multicols` environment and apply our “*vertical adjust*” spacing.

```

3565 \cs_new_protected:Nn \__enumext_multicols_stop:
3566 {
3567     \int_compare:nNt
3568     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3569     {
3570         \end{multicols}
3571         \__enumext_unskip_unkern:
3572         \__enumext_unskip_unkern:
3573         \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
3574     }
3575 }

```

(End of definition for `__enumext_multicols_stop:`)

`__enumext_after_list:` The function `__enumext_after_list:` first check the state of the boolean variable `\l__enumext_minipage_active_X_bool`, if it is “true” a small test will be executed to check if we have omitted the use of `\miniright` (the `__enumext_mini_page` environment has not been closed), then close `__enumext_mini_page` and add the *adjusted vertical space* `\l__enumext_minipage_after_skip`, otherwise we will close the `multicols` environment.

```

3576 \cs_new_protected:Nn \__enumext_after_list:
3577 {
3578     \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
3579     {
3580         \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
3581         {
3582             \msg_warning:nn { enumext } { missing-miniright }
3583             \miniright
3584         }
3585         \int_gzero:N \g__enumext_minipage_stat_int
3586         \__enumext_unskip_unkern: % remove topsep + [partopsep]
3587         \end__enumext_mini_page
3588     }
3589     {
3590         \__enumext_multicols_stop:
3591     }

```

Now we will execute the functions `__enumext_after_stop_list:` used by the key `after`, `__enumext_check_ans_key_hook:` used by the key `check-ans`, `__enumext_vspace_below:` used by the keys `below` and `below*`. Finally set `\l__enumext_standar_bool` to false and call the function `__enumext_resume_save_counter:` used by the `series`, `resume` and `resume*` keys.

```

3592 \__enumext_after_stop_list:
3593 \__enumext_check_ans_key_hook:
3594 \__enumext_vspace_below:
3595 \bool_set_false:N \l__enumext_standar_bool

```



```

3596     \__enumext_resume_save_counter:
3597 }

```

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`.

```

3598 \__enumext_after_env:nn {enumext} { \__enumext_execute_after_env: }

```

(End of definition for `__enumext_after_list:.`)

12.39 The environment `keyans`

The environment `keyans` also based on lists. The main differences with the `enumext` environment are the *nesting* and the way the *answers* (choice) will be stored and checked, this environment is intended exclusively for “multiple choice questions”.

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

```

3599 \NewDocumentEnvironment{keyans}{ 0{ } }
3600 {
3601     \__enumext_keyans_safe_exec:
3602     \__enumext_keyans_parse_keys:n {#1}
3603     \__enumext_before_list_v:
3604     \__enumext_start_list:nn
3605     { \tl_use:N \l__enumext_label_v_tl }
3606     {
3607         \__enumext_list_arg_two_v:
3608         \__enumext_before_keys_exec_v:
3609     }
3610     \__enumext_keyans_set_item_width:
3611     \__enumext_after_args_exec_v:
3612 }
3613 {
3614     \__enumext_check_starred_cmd:n { item }
3615     \__enumext_stop_list:
3616     \__enumext_after_list_v:
3617 }

```

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

`__enumext_keyans_set_item_width:`

The function `__enumext_keyans_set_item_width:` will set the value of `\itemwidth` taking into account the value established by the `list-offset` key.

```

3618 \cs_new_protected:Nn \__enumext_keyans_set_item_width:
3619 {
3620     \dim_set:Nn \itemwidth
3621     {
3622         \linewidth
3623     }
3624     \dim_compare:nT
3625     {
3626         \l__enumext_listoffset_v_dim != \c_zero_dim
3627     }
3628     {
3629         \dim_sub:Nn \itemwidth
3630         {
3631             \l__enumext_listoffset_v_dim
3632         }
3633     }
3634 }

```

(End of definition for `__enumext_keyans_set_item_width:.`)

`__enumext_keyans_safe_exec:`

The `keyans` environment will only be available if the `save-ans` key is active and can only be used at the “first level” within the `enumext` environment. We do not want the environment to be nested, so we will set a maximum at this point. If the conditions are not met, an error message will be returned.

```

3635 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3636 {
3637     \bool_if:NF \l__enumext_store_active_bool
3638     {
3639         \msg_error:nnnn { enumext } { wrong-place } { keyans } { save-ans }
3640     }
3641     \int_incr:N \l__enumext_keyans_level_int
3642     \bool_set_true:N \l__enumext_keyans_env_bool
3643     \__enumext_keyans_name_and_start:

```

```

3644 % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3645 \bool_set_false:N \l__enumext_store_active_bool
3646 \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
3647 {
3648     \msg_error:nn { enumext } { keyans-nested }
3649 }
3650 \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3651 {
3652     \msg_error:nn { enumext } { keyans-wrong-level }
3653 }
3654 }

```

(End of definition for \l__enumext_keyans_safe_exec:.)

```

\l__enumext_keyans_parse_keys:n Parse [key = val] for keyans environment.
3655 \cs_new_protected:Npn \l__enumext_keyans_parse_keys:n #1
3656 {
3657     \keys_set:nn { enumext / keyans } {#1}
3658 }

```

(End of definition for \l__enumext_keyans_parse_keys:n.)

\l__enumext_before_list_v: Same implementation as the one used in the enumext environment.

```

\l__enumext_keyans_multicols_start:
\l__enumext_keyans_multicols_stop:
\l__enumext_after_list_v:
3659 \cs_new_protected:Nn \l__enumext_before_list_v:
3660 {
3661     \l__enumext_vspace_above_v:
3662     \l__enumext_before_args_exec_v:
3663     \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3664     {
3665         \dim_set:Nn \l__enumext_minipage_left_v_dim
3666         {
3667             \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3668         }
3669         \bool_set_true:N \l__enumext_minipage_active_v_bool
3670         \int_gincr:N \g__enumext_minipage_stat_int
3671         \l__enumext_keyans_minipage_add_space:
3672         \l__enumext_mini_page{ \l__enumext_minipage_left_v_dim }
3673     }
3674     \l__enumext_keyans_multicols_start:
3675 }
3676 \cs_new_protected:Nn \l__enumext_keyans_multicols_stop:
3677 {
3678     \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3679     {
3680         \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3681         {
3682             \dim_set:Nn \l__enumext_columns_sep_v_dim
3683             {
3684                 (
3685                     \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3686                 ) / \l__enumext_columns_v_int
3687                 - \l__enumext_listoffset_v_dim
3688             }
3689         }
3690         \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim
3691         \dim_zero:N \columnseprule % no rule here
3692         \bool_if:NF \l__enumext_minipage_active_v_bool
3693         {
3694             \skip_zero:N \multicolsep
3695             \l__enumext_keyans_multi_addvspace:
3696         }
3697         \raggedcolumns
3698         \begin{multicols}{\l__enumext_columns_v_int}
3699     }
3700 }
3701 \cs_new_protected:Nn \l__enumext_keyans_multicols_stop:
3702 {
3703     \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3704     {
3705         \end{multicols}
3706         \l__enumext_unskip_unkern:

```

```

3707     \__enumext_unskip_unkern:
3708     \par\addvspace{ \l__enumext_multicols_below_v_skip }
3709   }
3710 }
3711 \cs_new_protected:Nn \__enumext_after_list_v:
3712 {
3713   \bool_if:NTF \l__enumext_minipage_active_v_bool
3714   {
3715     \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3716     {
3717       \msg_warning:nn { enumext } { missing-miniright }
3718       \miniright
3719     }
3720     \int_gzero:N \g__enumext_minipage_stat_int
3721     \__enumext_unskip_unkern: % remove topsep + [partopsep]
3722     \end__enumext_mini_page
3723     \par\addvspace{ \l__enumext_minipage_after_skip }
3724   }
3725   {
3726     \__enumext_keyans_multicols_stop:
3727   }
3728   \bool_set_false:N \l__enumext_keyans_env_bool
3729   \__enumext_after_stop_list_v:
3730   \__enumext_vspace_below_v:
3731 }

```

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

12.40 Tagging PDF support for non-standart list environments

The \TeX release 2022-06-01 brings automatic support for tagPDF in several aspects, including the standart *list environments* and the `list` environment. Unfortunately non-standard *list environments* like `keyanspic` or the horizontal list environments `enumext*` and `keyans*` are not structured in a nice way, i.e. the expected result in the PDF file is the expected one, but the underlying structure is not correct. In simple terms, for tagPDF a list environment is a list environment, no matter what it looks like in the PDF file.

To maintain a correct list structure when `\DocumentMetadata` is active, it is necessary to do some things manually. This implementation is an adaptation of my answer thanks to Ulrike Fischer's comments in [How can I modify my \item redefinition to be compatible with tagging-pdf.](#)

12.40.1 Socket for tagging support in `enumext*` and `keyans*`

We will first define the necessary sockets and their behavior for `enumext*` and `keyans*`.

```

start-list-tags
stop-start-tags
stop-list-tags
\__enumext_start_list_tag:n
\__enumext_stop_start_list_tag:
\__enumext_stop_list_tag:n
3732 \socket_new:nn {taggsupport/enumext/starred}{ 1 }
3733 \socket_new_plugin:nnn {taggsupport/enumext/starred} {start-list-tags}
3734 {
3735   \tag_resume:n {#1}
3736   \tag_struct_begin:n {tag=LI}
3737   \tag_struct_begin:n {tag=Lbl}
3738   \tag_mc_begin:n {tag=Lbl}
3739 }
3740 \socket_new_plugin:nnn {taggsupport/enumext/starred} {stop-start-tags}
3741 {
3742   \tag_mc_end:
3743   \tag_struct_end:n {tag=Lbl}
3744   \tag_struct_begin:n {tag=LBody}
3745   \tag_struct_begin:n {tag=text-unit}
3746   \tag_struct_begin:n {tag=text}
3747 }
3748 \socket_new_plugin:nnn {taggsupport/enumext/starred} {stop-list-tags}
3749 {
3750   \tag_struct_end:n {tag=text}
3751   \tag_struct_end:n {tag=text-unit}
3752   \tag_struct_end:n {tag=LBody}
3753   \tag_struct_end:n {tag=LI}
3754   \tag_suspend:n {#1}
3755 }

```

And now we'll wrap them so that they're only active when `\DocumentMetadata` is present.

```

3756 \cs_new_protected_nopar:Npn \__enumext_start_list_tag:n #1
3757 {
3758   \IfDocumentMetadataTF
3759   {
3760     \socket_assign_plugin:nn {taggsupport/enumext/starred} {start-list-tags}

```

```

3761         \socket_use:n {tagsupport/enumext/starred} {#1}
3762     }
3763     {}
3764 }
3765 \cs_new_protected_nopar:Nn \__enumext_stop_start_list_tag:
3766 {
3767     \IfDocumentMetadataTF
3768     {
3769         \socket_assign_plug:nn {tagsupport/enumext/starred} {stop-start-tags}
3770         \socket_use:nn {tagsupport/enumext/starred} { }
3771     }
3772     {}
3773 }
3774 \cs_new_protected_nopar:Npn \__enumext_stop_list_tag:n #1
3775 {
3776     \IfDocumentMetadataTF
3777     {
3778         \socket_assign_plug:nn {tagsupport/enumext/starred} {stop-list-tags}
3779         \socket_use:nn {tagsupport/enumext/starred} {#1}
3780     }
3781     {}
3782 }

```

(End of definition for *start-list-tags* and others.)

12.40.2 Socket for tagging support in keyanspic

start-list-tags We will first define the necessary sockets and their behavior for `keyanspic`.

```

stop-start-tags
stop-list-tags
\__enumext_anspic_start_list_tag:
\__enumext_anspic_stop_start_list_tag:
\__enumext_anspic_stop_list_tag:
3783 \socket_new:nn {tagsupport/enumext/keyanspic}{ 0 }
3784 \socket_new_plug:nnn {tagsupport/enumext/keyanspic} {start-list-tags}
3785 {
3786     \tag_resume:n {keyanspic}
3787     \tag_start:n {keyanspic}
3788     \tag_struct_begin:n {tag=LI}
3789     \tag_struct_begin:n {tag=Lbl}
3790     \tag_mc_begin:n {tag=Lbl}
3791 }
3792 \socket_new_plug:nnn {tagsupport/enumext/keyanspic} {stop-start-tags}
3793 {
3794     \tag_mc_end:
3795     \tag_struct_end:n {tag=Lbl}
3796     \tag_struct_begin:n {tag=LBody}
3797     \tag_struct_begin:n {tag=text-unit}
3798     \tag_struct_begin:n {tag=text}
3799     \tag_mc_begin:n {tag=text}
3800 }
3801 \socket_new_plug:nnn {tagsupport/enumext/keyanspic} {stop-list-tags}
3802 {
3803     \tag_mc_end:
3804     \tag_struct_end:n {tag=text-unit}
3805     \tag_struct_end:n {tag=text}
3806     \tag_struct_end:n {tag=LBody}
3807     \tag_struct_end:n {tag=LI}
3808     \tag_stop:n {keyanspic}
3809     \tag_suspend:n {keyanspic}
3810 }

```

And now we'll wrap them so that they're only active when `\DocumentMetadata` is present.

```

3811 \cs_new_protected_nopar:Nn \__enumext_anspic_start_list_tag:
3812 {
3813     \IfDocumentMetadataTF
3814     {
3815         \socket_assign_plug:nn {tagsupport/enumext/keyanspic} {start-list-tags}
3816         \socket_use:n {tagsupport/enumext/keyanspic}
3817     }
3818     {}
3819 }
3820 \cs_new_protected_nopar:Nn \__enumext_anspic_stop_start_list_tag:
3821 {
3822     \IfDocumentMetadataTF
3823     {
3824         \socket_assign_plug:nn {tagsupport/enumext/keyanspic} {stop-start-tags}
3825         \socket_use:nn {tagsupport/enumext/keyanspic}

```

```

3826     }
3827     {}
3828 }
3829 \cs_new_protected_nopar:Nn \__enumext_anspic_stop_list_tag:
3830 {
3831   \IfDocumentMetadataTF
3832   {
3833     \socket_assign_plug:nn {tagssupport/enumext/keyanspic} {stop-list-tags}
3834     \socket_use:nn {tagssupport/enumext/keyanspic}
3835   }
3836   {}
3837 }

```

(End of definition for `start-list-tags` and others.)

12.41 The environment `keyanspic` and `\anspic`

The `keyanspic` environment is a list-based environment that uses the same configuration for “*spacing*” and $\langle label \rangle$ 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 $\langle label \rangle$ underneath, adjusting widths according to the options passed to the environment.

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

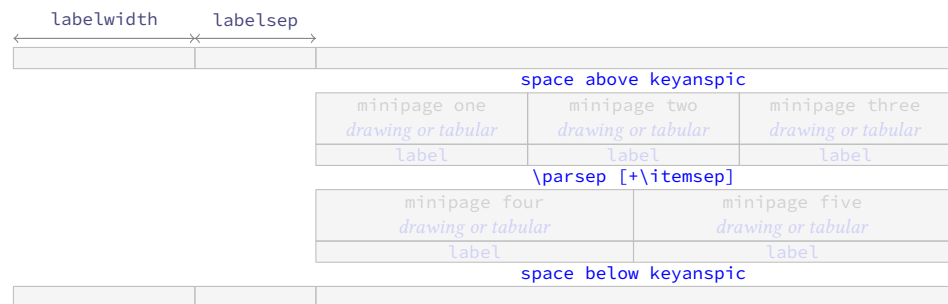


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

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

12.41.1 The environment `keyanspic`

The environment we wish to build will be based on the `list` environment and will take two optional arguments, the starred argument `*` will set the position of the $\langle label \rangle$ to the top if present and bottom otherwise, the second optional argument will process the number of *drawing* or *tabular* that will be on top, bottom or inline when present or not.

`__enumext_keyans_pic_safe_exec:n`

The function `__enumext_keyans_pic_safe_exec:n` check the starred argument and nested level position inside the `enumext` environment.

We will set the state of the variable `__enumext_keyans_pic_star_bool` along with the value of the variable `__enumext_keyans_pic_label_pos_str` according to the presence of the starred argument.

```

3838 \cs_new_protected:Npn \__enumext_keyans_pic_safe_exec:n #1
3839 {
3840   \int_incr:N \__enumext_keyans_pic_level_int
3841   \int_compare:nNnT { \__enumext_keyans_pic_level_int } > { 1 }
3842   {
3843     \msg_error:nn { enumext } { keyanspic-nested }
3844   }
3845   \__enumext_keyans_name_and_start:
3846   \bool_if:nTF { #1 }
3847   {
3848     \bool_set_true:N \__enumext_keyans_pic_star_bool
3849     \str_set:Nn \__enumext_keyans_pic_label_pos_str { t }
3850   }
3851   {
3852     \str_set:Nn \__enumext_keyans_pic_label_pos_str { b }
3853   }
3854 }

```

(End of definition for `__enumext_keyans_pic_safe_exec:n`.)

`__enumext_keyans_pic_skip_abs:N` The function `__enumext_keyans_pic_skip_abs:N` will return a positive value `\parsep`.

```

3855 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3856 {
3857   \dim_compare:nNnT { #1 } < { \c_zero_dim }
3858   {
3859     \skip_set:Nn #1 { -#1 }
3860   }
3861 }

```

(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.

```

3862 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
3863 {

```

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

```

3864   \bool_set_false:N \__enumext_leftmargin_tmp_v_bool
3865   \__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.

```

3866   %%%\skip_add:Nn \parsep { \itemsep }
3867   \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
3868   \dim_zero:N \labelwidth
3869   \dim_zero:N \listparindent
3870   \dim_zero:N \labelsep
3871   \skip_zero:N \partopsep
3872   \skip_zero:N \itemsep

```

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

```

3873   \__enumext_keyans_pic_skip_abs:N \parsep
3874   \skip_set:Nn \__enumext_keyans_pic_above_skip
3875   {
3876     \box_dp:N \strutbox
3877     + \__enumext_topsep_v_skip
3878     %%%- \parsep
3879   }
3880 }

```

(End of definition for `__enumext_keyans_pic_arg_two:`.)

`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.

```

3881 \NewDocumentEnvironment{keyanspic}{ s o }
3882 {
3883   \__enumext_keyans_pic_safe_exec:n { #1 }
3884   \begin{list} { } { \__enumext_keyans_pic_arg_two: }
3885   \IfDocumentMetadataTF
3886   {
3887     \tag_suspend:n {list}
3888   }{}
3889   \item[] \scan_stop:
3890   % paranoia
3891   \RenewDocumentCommand \item {}
3892   {
3893     \msg_error:nn { enumext } { keyanspic-item-cmd }
3894   }
3895   \IfDocumentMetadataTF
3896   {
3897     \tag_resume:n {keyanspic}
3898     \tag_tool:n {para/tagging=false}
3899     \tag_suspend:n {keyanspic}
3900   } { }

```

```

3901 \vspace { \l__enumext_keyans_pic_above_skip } % need see what this on space
3902 }

```

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.

```

3903 {
3904   \IfDocumentMetadataTF
3905   {
3906     \tag_start:n {keyanspic}
3907     \tag_struct_begin:n {tag=L,attribute=enumerate}
3908   }{ }
3909   \tl_if_novalue:nTF { #2 }
3910   {
3911     \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
3912   }
3913   { \__enumext_keyans_pic_do:n { #2 } }
3914   \IfDocumentMetadataTF { \tag_stop:n {keyanspic} } { }
3915   \end{list}
3916   \IfDocumentMetadataTF { \tag_struct_end: \tag_struct_end: } { }
3917   \__enumext_check_starred_cmd:n { anspic }
3918   \setcounter { enumXvi } { 0 }
3919   \par\addvspace{ \dim_eval:n { -\parsep + \box_dp:N \strutbox } }
3920   %\bool_set_false:N \l__enumext_store_active_bool
3921 }

```

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

```

\__enumext_keyans_pic_do:n
\__enumext_keyans_pic_do:e

```

The optional argument is split by comma and is handled directly by the function `__enumext_keyans_pic_do:n` and passed to the function `__enumext_keyans_pic_row:n`.

```

3922 \cs_new_protected:Nn \__enumext_keyans_pic_do:n
3923 {
3924   \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
3925 }
3926 \cs_generate_variant:Nn \__enumext_keyans_pic_do:n { e }

```

(End of definition for `__enumext_keyans_pic_do:n`.)

```

\__enumext_keyans_pic_row:n

```

The function `__enumext_keyans_pic_row:n` will set the widths for the `minipage` environments and place the content `<stored>` by `\anspic*` in the `\l__enumext_keyans_pic_body_seq` sequence inside them.

```

3927 \cs_new_protected:Nn \__enumext_keyans_pic_row:n
3928 {
3929   \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
3930   \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
3931   \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
3932   \int_step_inline:nnn
3933   { \l__enumext_keyans_pic_above_int + 1 }
3934   { \l__enumext_keyans_pic_below_int }
3935   {
3936     \IfDocumentMetadataTF { \tag_stop:n {minipage} } { } { }
3937     \begin{minipage}[ \l__enumext_keyans_pic_label_pos_str ]{ \l__enumext_keyans_pic_width_dim }
3938       \centering
3939       \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
3940     \end{minipage}
3941     \IfDocumentMetadataTF { \tag_start:n {minipage} } { } { }
3942   }
3943   \bool_if:NTF \l__enumext_keyans_pic_star_bool
3944   {
3945     \par
3946   }
3947   {
3948     \par%\vspace{ \box_ht_plus_dp:N \strutbox }
3949   }
3950 }

```

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

12.41.2 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.

```
3951 \NewDocumentCommand \anspic { s o +m }
3952 {
```

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

```
3953 \bool_if:NF \l__enumext_store_active_bool
3954 {
3955 \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
3956 }
3957 \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3958 {
3959 \msg_error:nn { enumext } { keyanspic-wrong-level }
3960 }
3961 \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
3962 {
3963 \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
3964 }
```

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.

```
3965 \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
3966 {
3967 \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
3968 }
3969 }
```

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

```
\__enumext_anspic_box_set_dim:n
\__enumext_keyans_anspic_label:nnn
\__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.

```
3970 \cs_new_protected:Npn \__enumext_anspic_box_set_dim:n #1
3971 {
3972 \vbox_set:Nn \l__enumext_anspic_body_box { #1 }
3973 \dim_set:Nn \l__enumext_anspic_body_htdp_dim
3974 {
3975 \box_ht_plus_dp:N \l__enumext_anspic_body_box
3976 }
3977 \vbox_set:Nn \l__enumext_anspic_label_box { \l__enumext_label_v_tl }
3978 \dim_set:Nn \l__enumext_anspic_label_htdp_dim
3979 {
3980 \box_ht_plus_dp:N \l__enumext_anspic_label_box
3981 }
3982 }
3983 % process label
3984 \cs_new_protected:Npn \__enumext_anspic_label:nn #1 #2
3985 {
3986 \bool_if:nT { #1 }
3987 {
3988 \__enumext_keyans_addto_prop:n { #2 }
3989 \__enumext_keyans_store_ref:
3990 \__enumext_keyans_addto_seq:n { #2 }
3991 \int_gincr:N \g__enumext_check_starred_cmd_int
3992 \bool_lazy_or:nnT
3993 { \bool_if_p:N \l__enumext_show_answer_bool }
3994 { \bool_if_p:N \l__enumext_show_position_bool }
3995 {
3996 \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
3997 \__enumext_keyans_show_left:n { #2 }
3998 \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
3999 }
4000 }
4001 \makebox[ \l__enumext_keyans_pic_width_dim ][ c ]
4002 {
4003 \tl_use:N \l__enumext_label_font_style_v_tl
4004 \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
4005 }
4006 }
```

```

4007 \cs_new_protected:Npn \__enumext_keyans_anspic_label:nnn #1 #2 #3
4008 {
4009   \stepcounter { enumXvi }
4010   \__enumext_anspic_box_set_dim:n { #3 }
4011   \bool_if:NTF \__enumext_keyans_pic_star_bool
4012   {
4013     \__enumext_anspic_label:nn { #1 } { #2 }
4014   }
4015   {
4016     \raisebox
4017     {
4018       -\dim_eval:n
4019       {
4020         \__enumext_anspic_label_htdp_dim
4021         + \__enumext_anspic_body_htdp_dim
4022         + \box_ht_plus_dp:N \strutbox
4023       }
4024     }
4025     [ opt ] [ opt ]
4026     {
4027       \__enumext_anspic_label:nn { #1 } { #2 }
4028     }
4029   }
4030 }
4031 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
4032 {
4033   \__enumext_anspic_start_list_tag:
4034   \__enumext_keyans_anspic_label:nnn { #1 } { #2 } { #3 }
4035   \__enumext_anspic_stop_start_list_tag:
4036   \\ #3
4037   \__enumext_anspic_stop_list_tag:
4038 }

```

(End of definition for `__enumext_anspic_box_set_dim:n`, `__enumext_keyans_anspic_label:nnn`, and `__enumext_keyans_anspic_code:nnn`.)

12.42 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, we have no other option than to define a cascade of functions.

12.42.1 Redefining `\footnote` command

```

\__enumext_footnotetext:nn
\__enumext_renew_footnote:
\__enumext_print_footnote:

```

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

```

4039 \cs_new_protected:Nn \__enumext_footnotetext:nn
4040 {
4041   \footnotetext[#1]{#2}
4042 }
4043 \cs_new_protected:Nn \__enumext_renew_footnote:
4044 {
4045   \seq_gclear:N \g__enumext_footnote_arg_seq
4046   \seq_gclear:N \g__enumext_footnote_int_seq
4047   \RenewDocumentCommand \footnote { o +m }
4048   {
4049     \tl_if_novalue:nTF {##1}
4050     {
4051       \stepcounter{footnote}
4052       \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
4053     }
4054     {
4055       \int_gset:Nn \g__enumext_footnote_int { ##1 }
4056     }
4057     \footnotemark [ \g__enumext_footnote_int ]
4058     \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
4059     \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
4060   }
4061 }
4062 \cs_new_protected:Nn \__enumext_print_footnote:
4063 {

```

```

4064 \seq_if_empty:NF \g__enumext_footnote_int_seq
4065 {
4066   \seq_map_pairwise_function:NNN
4067   \g__enumext_footnote_int_seq
4068   \g__enumext_footnote_arg_seq
4069   \__enumext_footnotetext:nn
4070 }
4071 }

```

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

12.42.2 Functions for item box width

To achieve the horizontal list environment we will capture the `\item` command and the $\langle content \rangle$ of this in *horizontal box* using `\makebox` for the `label` and a `minipage` environment for the $\langle content \rangle$ passed to `\item`, we will also add the optional argument ($\langle number \rangle$) to `\item` to be able to *join columns* horizontally, in simple terms, we want `\item` to behave in the same way as in the `enumext` environment but adding an optional first argument ($\langle number \rangle$).

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

```

\__enumext_starred_columns_set_vii:
\__enumext_starred_columns_set_viii:

```

```

4072 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
4073 {
4074   \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
4075   {
4076     \dim_set:Nn \l__enumext_columns_sep_vii_dim
4077     {
4078       ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
4079       / \l__enumext_columns_vii_int
4080     }
4081   }
4082   \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - 1 }
4083   \dim_set:Nn \l__enumext_item_width_vii_dim
4084   {
4085     ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
4086     / \l__enumext_columns_vii_int
4087     - \l__enumext_labelwidth_vii_dim
4088     - \l__enumext_labelsep_vii_dim
4089   }

```

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

```

4090   \dim_compare:nNnT { \l__enumext_rightmargin_vii_dim } > { \c_zero_dim }
4091   {
4092     \dim_sub:Nn \l__enumext_item_width_vii_dim
4093     {
4094       ( \l__enumext_rightmargin_vii_dim * \l__enumext_tmpa_vii_int )
4095       / \l__enumext_columns_vii_int
4096     }
4097     \dim_add:Nn \l__enumext_columns_sep_vii_dim
4098     {
4099       \l__enumext_rightmargin_vii_dim
4100     }
4101   }
4102 }

```

Same implementation for the `keyans*` environment.

```

4103 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
4104 {
4105   \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
4106   {
4107     \dim_set:Nn \l__enumext_columns_sep_viii_dim
4108     {
4109       ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
4110       / \l__enumext_columns_viii_int
4111     }
4112   }
4113   \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - 1 }
4114   \dim_set:Nn \l__enumext_item_width_viii_dim
4115   {
4116     ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
4117     / \l__enumext_columns_viii_int
4118     - \l__enumext_labelwidth_viii_dim
4119     - \l__enumext_labelsep_viii_dim
4120   }

```

```

4121 \dim_compare:nNnT { \l__enumext_rightmargin_viii_dim } > { \c_zero_dim }
4122 {
4123   \dim_sub:Nn \l__enumext_item_width_viii_dim
4124   {
4125     ( \l__enumext_rightmargin_viii_dim * \l__enumext_tmpa_vii_int )
4126     / \l__enumext_columns_viii_int
4127   }
4128   \dim_add:Nn \l__enumext_columns_sep_viii_dim
4129   {
4130     \l__enumext_rightmargin_viii_dim
4131   }
4132 }
4133 }

```

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

12.42.3 Functions for join item columns

`__enumext_starred_joined_item_vii:n`
`__enumext_starred_joined_item_viii:n`

The functions `__enumext_starred_joined_item_vii:n` and `__enumext_starred_joined_item_viii:n` will set the *width* of the box in which the *content* passed to `\item(<columns>)` will be stored together with the value of `\itemwidth` for the `enumext*` environment.

```

4134 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
4135 {
4136   \int_set:Nn \l__enumext_joined_item_vii_int {#1}
4137   \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
4138   {
4139     \msg_warning:nnee { enumext } { item-joined }
4140     { \int_use:N \l__enumext_joined_item_vii_int }
4141     { \int_use:N \l__enumext_columns_vii_int }
4142     \int_set:Nn \l__enumext_joined_item_vii_int
4143     {
4144       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4145     }
4146   }
4147   \int_compare:nNnT
4148   { \l__enumext_joined_item_vii_int }
4149   >
4150   { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4151   {
4152     \msg_warning:nnee { enumext } { item-joined-columns }
4153     { \int_use:N \l__enumext_joined_item_vii_int }
4154     {
4155       \int_eval:n
4156       { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4157     }
4158     \int_set:Nn \l__enumext_joined_item_vii_int
4159     {
4160       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4161     }
4162   }
4163   \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { 1 }
4164   {
4165     \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
4166     \int_decr:N \l__enumext_joined_item_aux_vii_int
4167     \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
4168     \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
4169     \dim_set:Nn \l__enumext_joined_width_vii_dim
4170     {
4171       \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
4172       + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
4173         + \l__enumext_columns_sep_vii_dim
4174       ) * \l__enumext_joined_item_aux_vii_int
4175     }
4176     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
4177   }
4178   {
4179     \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
4180     \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
4181   }
4182 }

```

Same implementation for the `keyans*` environment.

```

4183 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
4184 {
4185   \int_set:Nn \l__enumext_joined_item_viii_int {#1}
4186   \int_compare:nNt { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
4187   {
4188     \msg_warning:nnee { enumext } { item-joined }
4189     { \int_use:N \l__enumext_joined_item_viii_int }
4190     { \int_use:N \l__enumext_columns_viii_int }
4191     \int_set:Nn \l__enumext_joined_item_viii_int
4192     {
4193       \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4194     }
4195   }
4196   \int_compare:nNt
4197   { \l__enumext_joined_item_viii_int }
4198   >
4199   { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4200   {
4201     \msg_warning:nnee { enumext } { item-joined-columns }
4202     { \int_use:N \l__enumext_joined_item_viii_int }
4203     {
4204       \int_eval:n
4205       { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4206     }
4207     \int_set:Nn \l__enumext_joined_item_viii_int
4208     {
4209       \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4210     }
4211   }
4212   \int_compare:nNtF { \l__enumext_joined_item_viii_int } > { 1 }
4213   {
4214     \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
4215     \int_decr:N \l__enumext_joined_item_aux_viii_int
4216     \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
4217     \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
4218     \dim_set:Nn \l__enumext_joined_width_viii_dim
4219     {
4220       \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
4221       + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
4222         + \l__enumext_columns_sep_viii_dim
4223       ) * \l__enumext_joined_item_aux_viii_int
4224     }
4225     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
4226   }
4227   {
4228     \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
4229     \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
4230   }
4231 }

```

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

12.42.4 Functions for mini-env, mini-right and mini-right* keys

`__enumext_start_mini_vii:` The implementation of the `mini-env` key support is almost identical to the one used in the `enumext` and `keyans` environments, the difference is that the `__enumext_mini_page` environment on the “right side” is executed “after” closing the environment, so it is necessary to make a global copy of the variable `__enumext_minipage_right_vii_dim` in the variable `\g__enumext_minipage_right_vii_dim`.

```

4232 \cs_new_protected:Npn \__enumext_start_mini_vii:
4233 {
4234   \dim_compare:nNt { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
4235   {
4236     \dim_set:Nn \l__enumext_minipage_left_vii_dim
4237     {
4238       \linewidth
4239       - \l__enumext_minipage_right_vii_dim
4240       - \l__enumext_minipage_hsep_vii_dim
4241     }
4242     \bool_set_true:N \l__enumext_minipage_active_vii_bool
4243     \dim_gset_eq:NN
4244     \g__enumext_minipage_right_vii_dim
4245     \l__enumext_minipage_right_vii_dim

```

```

4246     \__enumext_mini_addvspace_vii:
4247     \nointerlineskip\noindent
4248     \__enumext_mini_page{ \l__enumext_minipage_left_vii_dim }
4249   }
4250 }

```

The function `__enumext_stop_mini_vii:` closes the `__enumext_mini_page` 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_page` on the “right side”.

```

4251 \cs_new_protected:Nn \__enumext_stop_mini_vii:
4252 {
4253   \bool_if:NT \l__enumext_minipage_active_vii_bool
4254   {
4255     \end__enumext_mini_page
4256     \hfill
4257     \bool_gset_true:N \g__enumext_minipage_active_vii_bool
4258   }
4259 }

```

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_page` 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.

```

4260 \__enumext_after_env:nn {enumext*}
4261 {
4262   \bool_if:NT \g__enumext_minipage_active_vii_bool
4263   {
4264     \__enumext_mini_page{ \g__enumext_minipage_right_vii_dim }
4265     \par\addvspace { \g__enumext_minipage_right_skip }
4266     \bool_if:NF \g__enumext_minipage_center_vii_bool
4267     {
4268       \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
4269       {
4270         \centering
4271       }
4272     }
4273     \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
4274     {
4275       \tl_use:N \g__enumext_miniright_code_vii_tl
4276     }
4277     \box_use_drop:N \l__enumext_miniright_code_vii_box
4278     \end__enumext_mini_page
4279     \par\addvspace{ \g__enumext_minipage_after_skip }
4280   }
4281   \bool_gset_false:N \g__enumext_minipage_active_vii_bool
4282   \bool_gset_true:N \g__enumext_minipage_center_vii_bool
4283   \tl_gclear:N \g__enumext_miniright_code_vii_tl
4284   \dim_gzero:N \g__enumext_minipage_right_vii_dim
4285   \bool_gset_false:N \g__enumext_starred_bool
4286 }

```

(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.

```

4287 \cs_new_protected:Nn \__enumext_start_mini_viii:
4288 {
4289   \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
4290   {
4291     \dim_set:Nn \l__enumext_minipage_left_viii_dim
4292     {
4293       \linewidth
4294       - \l__enumext_minipage_right_viii_dim
4295       - \l__enumext_minipage_hsep_viii_dim
4296     }
4297     \bool_set_true:N \l__enumext_minipage_active_viii_bool
4298     \dim_gset_eq:NN
4299       \g__enumext_minipage_right_viii_dim
4300       \l__enumext_minipage_right_viii_dim
4301     \__enumext_mini_addvspace_viii:

```

```

4302         \nointerlineskip\noindent
4303         \__enumext_mini_page{ \l__enumext_minipage_left_viii_dim }
4304     }
4305 }
4306 \cs_new_protected:Nn \__enumext_stop_mini_viii:
4307 {
4308     \bool_if:NT \l__enumext_minipage_active_viii_bool
4309     {
4310         \end__enumext_mini_page
4311         \hfill
4312         \bool_gset_true:N \g__enumext_minipage_active_viii_bool
4313     }
4314 }
4315 \__enumext_after_env:nn {keyans*}
4316 {
4317     \bool_if:NT \g__enumext_minipage_active_viii_bool
4318     {
4319         \__enumext_mini_page{ \g__enumext_minipage_right_viii_dim }
4320         \par\addvspace { \g__enumext_minipage_right_skip }
4321         \bool_if:NF \g__enumext_minipage_center_viii_bool
4322         {
4323             \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
4324             {
4325                 \centering
4326             }
4327         }
4328         \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
4329         {
4330             \tl_use:N \g__enumext_miniright_code_viii_tl
4331         }
4332         \box_use_drop:N \l__enumext_miniright_code_viii_box
4333         \end__enumext_mini_page
4334         \par\addvspace{ \g__enumext_minipage_after_skip }
4335     }
4336     \bool_gset_false:N \g__enumext_minipage_active_viii_bool
4337     \bool_gset_true:N \g__enumext_minipage_center_viii_bool
4338     \tl_gclear:N \g__enumext_miniright_code_viii_tl
4339     \dim_gzero:N \g__enumext_minipage_right_viii_dim
4340 }

```

(End of definition for __enumext_start_mini_viii: and __enumext_stop_mini_viii:.)

12.43 The environment enumext*

enumext* First we will generate the environment and we will give a temporary definition to __enumext_stop_item_tmp_vii: equal to __enumext_first_item_tmp_vii: and next to \item equal to __enumext_start_item_tmp_vii: which we will redefine later. Unlike the implementation used by the **shortlst** package, we will not set the values of \rightskip and \@rightskip equal to \@flushglue whose value is 0.0pt plus 1.0 fil, in the tests I have performed this fails in some circumstances and different results are obtained when using pdfTeX and LuaTeX.

```

4341 \NewDocumentEnvironment{enumext*}{o}
4342 {
4343     \__enumext_safe_exec_vii:
4344     \__enumext_parse_keys_vii:n {#1}
4345     \__enumext_before_list_vii:
4346     \__enumext_start_store_level_vii:
4347     \__enumext_start_list:nn { }
4348     {
4349         \__enumext_list_arg_two_vii:
4350         \__enumext_before_keys_exec_vii:
4351     }
4352     % Stop tagging
4353     \SuspendTagging{enumext*}
4354     \__enumext_starred_columns_set_vii:
4355     \item[] \scan_stop:
4356     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_first_item_tmp_vii:
4357     \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
4358     \ignorespaces
4359 }
4360 {
4361     % Close for first \item

```



```

4362     \IfDocumentMetadataTF { \tag_struct_end: } { }
4363     \__enumext_stop_item_tmp_vii:
4364     \__enumext_remove_extra_parsep_vii:
4365     \__enumext_stop_list:
4366     \__enumext_stop_store_level_vii:
4367     \__enumext_after_list_vii:
4368 }

```

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

`__enumext_safe_exec_vii:` We will first call the function `__enumext_internal_mini_page:` to create the environment `__enumext-mini_page`, then the function `__enumext_is_not_nested:` which sets `\g__enumext_starred_bool` to true if we are not nested within `enumext`, we will increment `\l__enumext_level_h_int` to restrict nesting of the environment, set `\l__enumext_starred_bool` to true and finally call the function `__enumext_is_on_first_level:` which sets `\l__enumext_starred_first_bool` to true if we are not nested, allowing the “storage system” to be used.

```

4369 \cs_new_protected:Nn \__enumext_safe_exec_vii:
4370 {
4371     \__enumext_internal_mini_page:
4372     \__enumext_is_not_nested:
4373     \int_incr:N \l__enumext_level_h_int
4374     \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
4375     {
4376         \msg_error:nn { enumext } { nested }
4377     }
4378     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
4379     {
4380         \msg_error:nnn { enumext } { nested-horizontal } { keyans* }
4381     }
4382     \bool_set_true:N \l__enumext_starred_bool
4383     \bool_set_false:N \l__enumext_standar_bool
4384     \__enumext_is_on_first_level:
4385 }

```

(End of definition for `__enumext_safe_exec_vii:.`)

`__enumext_parse_keys_vii:n` First we will clear the variable `\l__enumext_series_str` used by the key `series`, process the environment `[⟨key = val⟩]` and execute the function `__enumext_parse_series:n` and used by the key `series`, then we execute the function `__enumext_store_active_keys_vii:n` and reprocess the `⟨keys⟩` to pass them to the storage `⟨sequence⟩` if the key `save-key` is not active and finally we call the function `__enumext-nested_base_line_fix:` used by the key `base-fix`.

```

4386 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
4387 {
4388     \tl_if_novalue:nF {#1}
4389     {
4390         \str_clear:N \l__enumext_series_str
4391         \keys_set:nn { enumext / enumext* } {#1}
4392         \__enumext_parse_series:n {#1}
4393         \__enumext_store_active_keys_vii:n {#1}
4394         \__enumext_nested_base_line_fix:
4395     }
4396 }

```

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

`__enumext_before_list_vii:` The function `__enumext_before_list_vii:` first calls the function `__enumext_vspace_above_vii:` used by the keys `above` and `above*`, then calls the function `__enumext_check_ans_active:` for the check answer mechanism and finally calls the functions `__enumext_before_args_exec:` and `__enumext-start_mini_vii:` used by the keys `before*`, `mini-env`, `mini-right` and `mini-right*`.

```

4397 \cs_new_protected:Nn \__enumext_before_list_vii:
4398 {
4399     \__enumext_vspace_above_vii:
4400     \__enumext_check_ans_active:
4401     \__enumext_before_args_exec_vii:
4402     \__enumext_start_mini_vii:
4403 }

```

(End of definition for `__enumext_before_list_vii:.`)

`__enumext_after_list_vii:` The function `__enumext_after_list_vii:` first calls the function `__enumext_stop_mini_vii:` used by the keys `mini-env`, `mini-right` and `mini-right*`, then to the functions `__enumext_after_stop_list_vii:` used by the key `after`, `__enumext_check_ans_key_hook:` used by the key `check-ans`, `__enumext_vspace_below_vii:` used by the keys `below` and `below*`. Finally set `\l__enumext_starred_bool` to false and call the `__enumext_resume_save_counter:` function used by the `series`, `resume` and `resume*` keys.

```

4404 \cs_new_protected:Nn \__enumext_after_list_vii:
4405 {
4406   \__enumext_stop_mini_vii:
4407   \__enumext_after_stop_list_vii:
4408   \__enumext_check_ans_key_hook:
4409   \__enumext_vspace_below_vii:
4410   \bool_set_false:N \l__enumext_starred_bool
4411   \__enumext_resume_save_counter:
4412 }

```

(End of definition for `__enumext_after_list_vii:`.)

`__enumext_start_store_level_vii:` The `__enumext_start_store_level_vii:` and `__enumext_stop_store_level_vii:` functions activate the level saving mechanism for storage in `(sequence)` of the `\anskey` command and `anskey*` environment if `enumext*` are nested in `enumext`.

`__enumext_stop_store_level_vii:`

```

4413 \cs_new_protected:Nn \__enumext_start_store_level_vii:
4414 {
4415   \bool_if:NT \l__enumext_store_active_bool
4416   {
4417     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
4418     {
4419       \__enumext_store_level_open_vii:
4420     }
4421   }
4422 }
4423 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
4424 {
4425   \bool_if:NT \l__enumext_store_active_bool
4426   {
4427     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
4428     {
4429       \__enumext_store_level_close_vii:
4430     }
4431   }
4432 }

```

(End of definition for `__enumext_start_store_level_vii:` and `__enumext_stop_store_level_vii:`.)

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

`__enumext_first_item_tmp_vii:` The `__enumext_first_item_tmp_vii:` function will remove horizontal space equal to `\labelwidth` plus `\labelsep` to the left of the first `\item` in the environment at the point of execution of this function, where it is equal to the `__enumext_stop_item_tmp_vii:` function inside the environment body definition.

```

4433 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_vii:
4434 {
4435   \skip_horizontal:n { -\l__enumext_labelwidth_vii_dim - \l__enumext_labelsep_vii_dim }
4436 }

```

(End of definition for `__enumext_first_item_tmp_vii:`.)

`__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`.

```

4437 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
4438 {
4439   \__enumext_stop_item_tmp_vii:
4440   \int_incr:N \l__enumext_item_column_pos_vii_int
4441   \int_gincr:N \g__enumext_item_count_all_vii_int
4442   \__enumext_item_peek_args_vii:
4443 }

```

(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).

```

4444 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
4445 {
4446   \peek_meaning:NTF (
4447     { \__enumext_joined_item_vii:w }
4448     { \__enumext_joined_item_vii:w (1) }
4449   }

```

(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`.

```

4450 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
4451 {
4452   \__enumext_starred_joined_item_vii:n {#1}
4453   \peek_meaning_remove:NTF *
4454   { \__enumext_starred_item_vii:w }
4455   { \__enumext_standar_item_vii:w }
4456 }

```

(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]`.

```

4457 \cs_new_protected:Npn \__enumext_standar_item_vii:w
4458 {
4459   \bool_set_false:N \l__enumext_item_starred_vii_bool
4460   \peek_meaning:NTF [
4461   {
4462     \bool_set_eq:NN \l__enumext_wrap_label_vii_bool \l__enumext_wrap_label_opt_vii_bool
4463     \__enumext_start_item_vii:w
4464   }
4465   {
4466     \bool_set_true:N \l__enumext_wrap_label_vii_bool
4467     \legacy_if_set_true:n { @noitemarg }
4468     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4469   }
4470 }

```

(End of definition for `__enumext_standar_item_vii:w`.)

`__enumext_starred_item_vii:w` The function `__enumext_starred_item_vii:w` together with the specified auxiliary functions `aux_i:w`, `aux_ii:w`, and `aux_iii:w` execute `\item*`, `\item* [<symbol>]` and `\item* [<symbol>] [<offset>]`.

```

4471 \cs_new_protected:Npn \__enumext_starred_item_vii:w
4472 {
4473   \bool_set_true:N \l__enumext_item_starred_vii_bool
4474   \bool_set_true:N \l__enumext_wrap_label_vii_bool
4475   \peek_meaning:NTF [
4476   { \__enumext_starred_item_vii_aux_i:w }
4477   { \__enumext_starred_item_vii_aux_ii:w }
4478   }
4479 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
4480 {
4481   \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
4482   \__enumext_starred_item_vii_aux_ii:w
4483 }
4484 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
4485 {
4486   \peek_meaning:NTF [
4487   { \__enumext_starred_item_vii_aux_iii:w }

```

```

4488     {
4489         \dim_set_eq:NN \l__enumext_item_symbol_sep_vii_dim \l__enumext_labelsep_vii_dim
4490         \legacy_if_set_true:n { @noitemarg }
4491         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4492     }
4493 }
4494 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
4495 {
4496     \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
4497     \legacy_if_set_true:n { @noitemarg }
4498     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4499 }

```

(End of definition for `__enumext_starred_item_vii:w` and others.)

`__enumext_fake_make_label_vii:n`

The `__enumext_fake_make_label_vii:n` function will be in charge of handling our definition of `\item`. First we increment the counter `enumXvii` for the enumerated items and activate support for the *check answers* mechanism, followed by support for `\item*[\langle symbol \rangle][\langle offset \rangle]` if present, then the `wrap-label` and `wrap-label*` keys which we execute using `\makebox` whose width will be given by the `labelwidth` key and position by the `align` key, inside the argument of this we will execute the `font` key together with the function defined by the `wrap-label` or `wrap-label*` keys. Finally we execute the `labelsep` key applying a *horizontal space*.

```

4500 \cs_new_protected_nopar:Npn \__enumext_fake_make_label_vii:n #1
4501 {
4502     \legacy_if:nT { @noitemarg }
4503     {
4504         \legacy_if_set_false:n { @noitemarg }
4505         \legacy_if:nT { @nmbrrlist }
4506         {
4507             \refstepcounter{enumXvii}
4508             \bool_if:NT \l__enumext_check_answers_bool
4509             {
4510                 \int_gincr:N \g__enumext_item_number_int
4511                 \bool_set_true:N \l__enumext_item_number_bool
4512             }
4513         }
4514     }
4515     \bool_if:NT \l__enumext_item_starred_vii_bool
4516     {
4517         \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
4518         {
4519             \tl_gset_eq:NN
4520             \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
4521         }
4522         \mode_leave_vertical:
4523         \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
4524         \hbox_overlap_left:n { \g__enumext_item_symbol_aux_vii_tl }
4525         \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
4526         \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
4527     }
4528     \bool_if:NTF \l__enumext_wrap_label_vii_bool
4529     {
4530         \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
4531         {
4532             \tl_use:N \l__enumext_label_font_style_vii_tl
4533             \__enumext_wrapper_label_vii:n {#1}
4534         }
4535     }
4536     {
4537         \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
4538         {
4539             \tl_use:N \l__enumext_label_font_style_vii_tl #1
4540         }
4541     }
4542     \skip_horizontal:N \l__enumext_labelsep_vii_dim
4543 }

```

(End of definition for `__enumext_fake_make_label_vii:n`.)

12.43.2 Real definition of \item in enumext*

The functions `__enumext_start_item_vii:w` and `__enumext_stop_item_vii:` executing the true definition of `\item` inside the `enumext*` environment, unlike the implementation in `shortlst` we will NOT use an extra group and the plain form of the `lrbox` environment.

`__enumext_start_item_vii:w`

The first thing we will do is set the value of `__enumext_stop_item_tmp_vii:` equal to `__enumext_stop_item_vii:` which we will define later, after that we will start capturing `\item` and its `⟨contents⟩` in a *horizontal box* where the width will be `\itemwidth` plus `\labelwidth` plus `\labelsep`.

```
4544 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
4545 {
4546   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
4547   \hbox_set_to_wd:Nnw \l__enumext_item_text_vii_box
4548   {
4549     \l__enumext_joined_width_vii_dim
4550     + \l__enumext_labelwidth_vii_dim
4551     + \l__enumext_labelsep_vii_dim
4552   }
```

If `\DocumentMetadata` is not active and the state of the variable `\l__enumext_footnotes_key_bool` is false, we will redefine the `\footnote` command.

```
4553   \IfDocumentMetadataTF { }
4554   {
4555     \bool_if:NF \l__enumext_footnotes_key_bool
4556     {
4557       \__enumext_renew_footnote:
4558     }
4559   }
```

Now we insert our *sockets* for the tagPDF support and print `\item`.

```
4560   \__enumext_start_list_tag:n {enumext*}
4561   \__enumext_fake_make_label_vii:n {#1}
4562   \__enumext_stop_start_list_tag:
```

Finally we open the `minipage` environment capture the `⟨item content⟩` and execute the `first` key, `listparindent` key which will be equal to `\parindent`, the `parsep` key which will be equal to `\parskip` and the `itemindent` key.

```
4563   \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
4564   \tl_use:N \l__enumext_after_list_args_vii_tl
4565   \dim_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
4566   \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
4567   \tl_use:N \l__enumext_fake_item_indent_vii_tl
4568 }
```

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

`__enumext_stop_item_vii:`

The `__enumext_stop_item_vii:` function will finish the fetching `\item` and its `⟨content⟩` by closing the `minipage` environment, the *sockets* for the tagPDF and the *horizontal box*.

```
4569 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
4570 {
4571   \__enumext_endminipage:
4572   \__enumext_stop_list_tag:n {enumext*}
4573   \hbox_set_end:
```

Here we will reduce the *warnings* a bit by setting the value of `\hbadness` to `10000`, print the `⟨contents⟩` of the *box* along with `\footnote`.

```
4574   \int_set:Nn \hbadness { 10000 }
4575   \box_use_drop:N \l__enumext_item_text_vii_box
4576   \IfDocumentMetadataTF { }
4577   {
4578     \bool_if:NF \l__enumext_footnotes_key_bool
4579     {
4580       \__enumext_print_footnote:
4581     }
4582   }
```

Finally set the *vertical* and *horizontal* spaces between rows and columns.

```
4583   \int_compare:nNnTF
4584   { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
4585   {
4586     \par\noindent
4587     \int_zero:N \l__enumext_item_column_pos_vii_int
4588   }
```

```

4589     {
4590         \skip_horizontal:N \l__enumext_columns_sep_vii_dim
4591     }
4592 }

```

(End of definition for __enumext_stop_item_vii:.)

__enumext_remove_extra_parsep_vii:

Finally we will remove the vertical space equal to `\parsep=\itemsep` when the total number of items is divisible by the number of items in the last row of the environment. Here the use of `\unskip` or `\removeatlastskip` fails and does not obtain the expected result, using `\vspace` is the option and in this case, we can use a simplified version since we are always in *vertical mode*.

```

4593 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
4594 {
4595     \int_compare:nNnT
4596     {
4597         \int_mod:nn
4598         { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
4599     }
4600     =
4601     { 0 }
4602     {
4603         \para_end:
4604         \skip_vertical:n { -\l__enumext_itemsep_vii_skip }
4605         \skip_vertical:N \c_zero_skip
4606         \int_gzero:N \g__enumext_item_count_all_vii_int
4607     }
4608 }

```

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`.

```

4609 \__enumext_after_env:nn {enumext*} { \__enumext_execute_after_env: }

```

(End of definition for __enumext_remove_extra_parsep_vii:.)

12.44 The environment keyans*

keyans*

First we will generate the environment and we will give a temporary definition to `__enumext_stop_item_tmp_viii:` equal to `__enumext_first_item_tmp_viii:` and next to `\item` equal to `__enumext_start_item_tmp_viii:` which we will redefine later. The implementation of this environment is the same as that used by the `enumext*` environment except for the `__enumext_check_starred_cmd:n` function added in the second part.

```

4610 \NewDocumentEnvironment{keyans*}{o}{
4611     {
4612         \__enumext_safe_exec_viii:
4613         \__enumext_parse_keys_viii:n {#1}
4614         \__enumext_before_list_viii:
4615         \__enumext_start_list:nn { }
4616         {
4617             \__enumext_list_arg_two_viii:
4618             \__enumext_before_keys_exec_viii:
4619         }
4620         \__enumext_starred_columns_set_viii:
4621         \item[] \scan_stop:
4622         \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_first_item_tmp_viii:
4623         \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
4624         \ignorespaces
4625     }
4626     {
4627         \__enumext_stop_item_tmp_viii:
4628         \__enumext_remove_extra_parsep_viii:
4629         \__enumext_check_starred_cmd:n { item }
4630         \__enumext_stop_list:
4631         \__enumext_after_list_viii:
4632     }
}

```

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

__enumext_safe_exec_viii:

The `__enumext_safe_exec_viii:` function will first check if the `save-ans` key is active and only when this is true the environment will be available, it will increment the value of `\l__enumext_keyans_level_h_int` and return an error message when we are nesting the environment, then it will call the `__enumext_keyans_name_and_start:` function in charge of saving the name of the environment and the line it is

running on, then it will check if we are trying to nest `keyans*` in `enumext*` returning an error and we will set `__enumext_starred_bool` to true, finally we will check if we are within the appropriate level within the `enumext` environment.

```

4633 \cs_new_protected:Nn \__enumext_safe_exec_viii:
4634 {
4635   \bool_if:NF \__enumext_store_active_bool
4636   {
4637     \msg_error:nnnn { enumext } { wrong-place } { keyans* } { save-ans }
4638   }
4639   \int_incr:N \__enumext_keyans_level_h_int
4640   \int_compare:nNnT { \__enumext_keyans_level_h_int } > { 1 }
4641   {
4642     \msg_error:nn { enumext } { nested }
4643   }
4644   \__enumext_keyans_name_and_start:
4645   \bool_if:NT \__enumext_starred_bool
4646   {
4647     \msg_error:nnn { enumext } { nested-horizontal } { enumext* }
4648   }
4649   \bool_set_true:N \__enumext_starred_bool
4650   % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
4651   \bool_set_false:N \__enumext_store_active_bool
4652   \int_compare:nNnT { \__enumext_level_int } > { 1 }
4653   {
4654     \msg_error:nn { enumext } { keyans-wrong-level }
4655   }
4656 }

```

(End of definition for `__enumext_safe_exec_viii:`)

```

\__enumext_parse_keys_viii:n Parse [⟨key = val⟩] for keyans*.
4657 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
4658 {
4659   \tl_if_novalue:nF {#1}
4660   {
4661     \keys_set:nn { enumext / keyans* } {#1}
4662   }
4663 }

```

(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`.

```

4664 \cs_new_protected:Nn \__enumext_before_list_viii:
4665 {
4666   \__enumext_vspace_above_viii:
4667   \__enumext_before_args_exec_viii:
4668   \__enumext_start_mini_viii:
4669 }

```

(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.

```

4670 \cs_new_protected:Nn \__enumext_after_list_viii:
4671 {
4672   \__enumext_stop_mini_viii:
4673   \__enumext_after_stop_list_viii:
4674   \__enumext_vspace_below_viii:
4675 }

```

(End of definition for `__enumext_after_list_viii:`)

12.44.1 The command `\item` in `keyans*`

The idea here is to make the `\item` command behave in the same way as in the `keyans` environment with the difference of the optional argument (`<number>`) which works in the same way as in the `enumext*` environment. In simple terms we want to store the `<label>` next to the [`<content>`] if it is present in the `<sequence>` and `<prop list>` defined by `save-ans` key for `\item*`, `\item* [<content>]`, `\item(<number>)*` and `\item(<number>)* [<content>]` commands.

`__enumext_first_item_tmp_viii:`

The `__enumext_first_item_tmp_viii:` function will remove horizontal space equal to `\labelwidth` plus `\labelsep` to the left of the first `\item` in the environment at the point of execution of this function, where it is equal to the `__enumext_stop_item_tmp_viii:` function inside the environment body definition.

```
4676 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_viii:
4677 {
4678   \skip_horizontal:n { -\__enumext_labelwidth_viii_dim - \__enumext_labelsep_viii_dim }
4679 }
```

(End of definition for `__enumext_first_item_tmp_viii:`)

`__enumext_start_item_tmp_viii:`

First we will call the function `__enumext_stop_item_tmp_viii:` that we will redefine later, we will increment the value of `\l__enumext_item_column_pos_viii_int` that will count the item's by rows and the value of `\g__enumext_item_count_all_viii_int` that will count the total of item's in the environment. After that we will call the function `__enumext_item_peek_args_viii:` that will handle the arguments passed to `\item`.

```
4680 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
4681 {
4682   \__enumext_stop_item_tmp_viii:
4683   \int_incr:N \l__enumext_item_column_pos_viii_int
4684   \int_gincr:N \g__enumext_item_count_all_viii_int
4685   \__enumext_item_peek_args_viii:
4686 }
```

(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).

```
4687 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
4688 {
4689   \peek_meaning:NTF (
4690     { \__enumext_joined_item_viii:w }
4691     { \__enumext_joined_item_viii:w (1) }
4692 }
```

(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`.

```
4693 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
4694 {
4695   \__enumext_starred_joined_item_viii:n {#1}
4696   \peek_meaning_remove:NTF *
4697     { \__enumext_starred_item_viii:w }
4698     { \__enumext_standar_item_viii:w }
4699 }
```

(End of definition for `__enumext_joined_item_viii:w`)

`__enumext_standar_item_viii:w`

The function `__enumext_standar_item_viii:w` will first look for the argument “[”, if present it will set the state of the variable `\l__enumext_wrap_label_opt_viii_bool` equal to the state of the variable `\l__enumext_wrap_label_opt_viii_bool` handled by the key `wrap-label*` and finally execute the *non-enumerated* version `\item [<custom>]` by means of the function `__enumext_start_item_viii:w`, otherwise we will set the value of the variable `\l__enumext_wrap_label_viii_bool` handled by the `wrap-label` key to true and set the switch `\if@noitemarg` to true to execute the *enumerated* version of `\item` by means of the function `__enumext_start_item_viii:w [__enumext_label_viii_tl]`.

```
4700 \cs_new_protected:Npn \__enumext_standar_item_viii:w
4701 {
4702   \bool_set_false:N \l__enumext_item_starred_viii_bool
4703   \peek_meaning:NTF [
```

```

4704     {
4705         \bool_set_eq:NN \l__enumext_wrap_label_viii_bool \l__enumext_wrap_label_opt_viii_bool
4706         \__enumext_start_item_viii:w
4707     }
4708     {
4709         \bool_set_true:N \l__enumext_wrap_label_viii_bool
4710         \legacy_if_set_true:n { @noitemarg }
4711         \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4712     }
4713 }

```

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

```

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

```

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

```

4714 \cs_new_protected:Npn \__enumext_starred_item_viii:w
4715 {
4716     \bool_set_true:N \l__enumext_item_starred_viii_bool
4717     \bool_set_true:N \l__enumext_wrap_label_viii_bool
4718     \peek_meaning:NTF [
4719         { \__enumext_starred_item_viii_aux_i:w }
4720         { \__enumext_starred_item_viii_aux_ii:w }
4721     }

```

The function `__enumext_starred_item_viii_aux_i:w` will save the optional argument to `\item*` in `\l__enumext_store_current_opt_arg_tl` and will save this argument along with the spacing set by the key `save-sep` in variable `\l__enumext_store_current_label_tl` if present, then call the function `__enumext_starred_item_viii_aux_ii:w`.

```

4722 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
4723 {
4724     \tl_clear:N \l__enumext_store_current_label_tl
4725     \tl_if_no_value:nF { #1 }
4726     {
4727         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
4728         {
4729             \tl_put_right:Ne \l__enumext_store_current_label_tl
4730             {
4731                 \l__enumext_store_keyans_item_opt_sep_tl
4732             }
4733             \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
4734         }
4735         \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
4736     }
4737     \__enumext_starred_item_viii_aux_ii:w
4738 }
4739 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
4740 {
4741     \legacy_if_set_true:n { @noitemarg }
4742     \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4743 }

```

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

```
\__enumext_starred_item_exec:
```

The function `__enumext_starred_item_exec:` will be in charge of storing the current `\label` for `\item*` followed by the `[\langle content \rangle]` for `\item*[\langle content \rangle]` if present in the `\sequence` and `\prop list` set by the `save-ans` key. In this same function the keys `show-ans`, `show-pos` and `save-ref` are implemented.

```

4744 \cs_new_protected:Nn \__enumext_starred_item_exec:
4745 {
4746     \tl_put_left:Ne \l__enumext_store_current_label_tl { \l__enumext_label_viii_tl }
4747     \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
4748     \__enumext_keyans_store_ref:
4749     \tl_put_left:Ne \l__enumext_store_current_label_tl { \item }
4750     \__enumext_keyans_addto_seq_link:
4751     \int_gincr:N \g__enumext_check_starred_cmd_int
4752     \bool_if:NT \l__enumext_show_answer_bool
4753     {
4754         \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4755     }
4756     \bool_if:NT \l__enumext_show_position_bool
4757     {

```

```

4758     \tl_set:Nx \l__enumext_mark_answer_sym_tl
4759     {
4760         \group_begin:
4761         \exp_not:N \normalfont
4762         \exp_not:N \footnotesize [ \int_eval:n
4763         {
4764             \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
4765         }
4766         ]
4767         \group_end:
4768     }
4769     \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4770 }
4771 }

```

(End of definition for `__enumext_starred_item_exec:`.)

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

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

```

4772 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
4773 {
4774     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
4775     \legacy_if:nT { @noitemarg }
4776     {
4777         \legacy_if_set_false:n { @noitemarg }
4778         \legacy_if:nT { @nmbrlist }
4779         {
4780             \bool_if:NT \l__enumext_hyperref_bool
4781             {
4782                 \legacy_if_set_true:n { @hyper@item }
4783             }
4784             \refstepcounter{enumXviii}
4785         }
4786     }

```

Here we start capturing `\item` and its *contents* in a *horizontal box*.

```

4787     \hbox_set_to_wd:Nnw \l__enumext_item_text_viii_box
4788     {
4789         \l__enumext_joined_width_viii_dim
4790         + \l__enumext_labelwidth_viii_dim
4791         + \l__enumext_labelsep_viii_dim
4792     }
4793     \bool_if:NF \l__enumext_footnotes_key_bool
4794     {
4795         \__enumext_renew_footnote:
4796     }
4797     \bool_if:NT \l__enumext_item_starred_viii_bool
4798     {
4799         \__enumext_starred_item_exec:
4800     }
4801     \bool_if:NTF \l__enumext_wrap_label_viii_bool
4802     {
4803         \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
4804         {
4805             \tl_use:N \l__enumext_label_font_style_viii_tl
4806             \__enumext_wrapper_label_viii:n {#1}
4807         }
4808     }
4809     {
4810         \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
4811         {
4812             \tl_use:N \l__enumext_label_font_style_viii_tl #1
4813         }
4814     }
4815     }
4816     \skip_horizontal:N \l__enumext_labelsep_viii_dim
4817     \tl_use:N \l__enumext_after_list_args_viii_tl
4818     \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
4819     \dim_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
4820     \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
4821     \bool_if:NT \l__enumext_item_starred_viii_bool

```

```

4822         {
4823             \tl_use:N \l__enumext_fake_item_indent_viii_tl
4824             \__enumext_keyans_show_item_opt:
4825             \skip_horizontal:n { -\l__enumext_fake_item_indent_viii_dim - \l__enumext_labelsep_
4826         }
4827         {
4828             \tl_use:N \l__enumext_fake_item_indent_viii_tl
4829         }
4830 %% First attempt here, need stop tag and set counter for enumi, and more...need test
4831 %% \RenewDocumentCommand \item { o }
4832 %% {
4833 %%     \tl_if_novalue:nTF {#1}
4834 %%     {
4835 %%         \__enumext_item_std:w
4836 %%     }
4837 %%     {
4838 %%         \__enumext_item_std:w [#1]
4839 %%     }
4840 %% }
4841 }

```

(End of definition for __enumext_start_item_viii:w.)

__enumext_stop_item_viii: The __enumext_stop_item_viii: function will finish the fetching \item and its *content* by closing the minipage environment and the horizontal box. Here we will reduce the warnings a bit by setting the value of \hbadness to 10000, print the *contents* of the box along with \footnote and finally set the vertical and horizontal spaces between rows and columns.

```

4842 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
4843 {
4844     \__enumext_endminipage:
4845     \hbox_set_end:
4846     \int_set:Nn \hbadness { 10000 }
4847     \box_use_drop:N \l__enumext_item_text_viii_box
4848     \bool_if:NF \l__enumext_footnotes_key_bool
4849     {
4850         \__enumext_print_footnote:
4851     }
4852     \int_compare:nNnTF
4853     { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
4854     {
4855         \par\noindent
4856         \int_zero:N \l__enumext_item_column_pos_viii_int
4857     }
4858     {
4859         \skip_horizontal:N \l__enumext_columns_sep_viii_dim
4860     }
4861 }

```

(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.

```

4862 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
4863 {
4864     \int_compare:nNnT
4865     {
4866         \int_mod:nn
4867         { \g__enumext_item_count_all_viii_int }
4868         { \l__enumext_columns_viii_int }
4869     }
4870     =
4871     { 0 }
4872     {
4873         \para_end:
4874         \skip_vertical:n { -\l__enumext_itemsep_viii_skip }
4875         \skip_vertical:N \c_zero_skip
4876         \int_gzero:N \g__enumext_item_count_all_viii_int
4877     }
4878 }

```

(End of definition for __enumext_remove_extra_parsep_viii:.)

12.45 The command \getkeyans

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

```

4879 \NewDocumentCommand \getkeyans { m }
4880 {
4881   \exp_args:Ne \__enumext_getkeyans_aux:n
4882   { \tl_to_str:e { \text_expand:n {#1} } }
4883 }

```

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

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

```

4884 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
4885 {
4886   \str_if_in:nnTF {#1} { : }
4887   {
4888     \use:e
4889     {
4890       \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
4891       { {##1} {##2} }
4892     }
4893     \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
4894   }
4895   { \msg_error:nnn { enumext } { missing-colon } {#1} }
4896 }

```

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

`__enumext_getkeyans:nn` The internal function `__enumext_getkeyans:nn` will check for the existence of the $\langle prop\ list \rangle$, if it does not exist it will return an error message, then it will fetch the content specified by the second $\langle argument \rangle$ from $\langle prop\ list \rangle$.

```

4897 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
4898 {
4899   \prop_if_exist:cTF { g__enumext_#1_prop }
4900   {
4901     \prop_item:cn { g__enumext_#1_prop } {#2}
4902   }
4903   {
4904     \msg_error:nnn { enumext } { undefined-storage-anskey } {#1}
4905   }
4906 }

```

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

12.46 The command \printkeyans

The `\printkeyans` command prints “all stored content” in the $\langle sequence \rangle$ defined by the `save-ans` key. The first thing we will do is define a set of $\langle filtered\ keys \rangle$ 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 $\langle keys \rangle$ for `\printkeyans*` and will be set by `\setenumext[$\langle print^* \rangle$]` and the variable `\l__enumext_print_keyans_vii_tl` will have the default keys for the environment `enumext*` nested within the $\langle sequence \rangle$ and will be set by `\setenumext[$\langle print, * \rangle$]`, the rest of the variables will be for the environment `enumext` and will be set by `\setenumext[$\langle print, level \rangle$]`.

```

4907 \keys_define:nn { enumext / print }
4908 {
4909   print* .code:n = \keys_precompile:neN { enumext / enumext* }
4910             { \__enumext_filter_save_key:n {#1} }
4911             \l__enumext_print_keyans_starred_tl, % starred cmd
4912   print* .initial:n = { nosep, label=\arabic*, columns=2, first=\small, font=\small },
4913   print-1 .code:n = \keys_precompile:neN { enumext / level-1 }
4914             { \__enumext_filter_save_key:n {#1} }
4915             \l__enumext_print_keyans_i_tl,
4916   print-1 .initial:n = { nosep, label=\arabic*, columns=2, first=\small, font=\small },
4917   print-2 .code:n = \keys_precompile:neN { enumext / level-2 }
4918             { \__enumext_filter_save_key:n {#1} }
4919             \l__enumext_print_keyans_ii_tl,
4920   print-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },

```

```

4921   print-3 .code:n      = \keys_precompile:neN { enumext / level-3 }
4922                               { \__enumext_filter_save_key:n {#1} }
4923                               \l__enumext_print_keyans_iii_tl,
4924   print-3 .initial:n    = { nosep, label=\roman*., first=\small, font=\small },
4925   print-4 .code:n      = \keys_precompile:neN { enumext / level-4 }
4926                               { \__enumext_filter_save_key:n {#1} }
4927                               \l__enumext_print_keyans_iv_tl,
4928   print-4 .initial:n    = { nosep, label=\Alph*., first=\small, font=\small },
4929   print-* .code:n      = \keys_precompile:neN { enumext / enumext* }
4930                               { \__enumext_filter_save_key:n {#1} }
4931                               \l__enumext_print_keyans_vii_tl, % starred nested
4932   print-* .initial:n    = { nosep, label=\arabic*., first=\small, font=\small },
4933   }

```

• The reason for storing *(keys)* in token lists using `\keys_precompile:neN` is because the keys are set via `\setenumext` but are later executed by running the command `\printkeyans` and they are not handled directly by its optional argument, except those related to the first opening level.

`\printkeyans` Create a user command to print “all stored content” in *(sequence)* for `\anskey`, `anskey*`, `\item*` and `\anspic*`. Within a group we will run our “precompiled keys” and then call the internal function `__enumext_printkeyans:nnn`.

```

4934 \NewDocumentCommand \printkeyans { s O{} m }
4935 {
4936   \group_begin:
4937     \tl_use:N \l__enumext_print_keyans_i_tl
4938     \tl_use:N \l__enumext_print_keyans_ii_tl
4939     \tl_use:N \l__enumext_print_keyans_iii_tl
4940     \tl_use:N \l__enumext_print_keyans_iv_tl
4941     \tl_use:N \l__enumext_print_keyans_vii_tl
4942     \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
4943   \group_end:
4944 }

```

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

`__enumext_printkeyans:nnn` The internal function `__enumext_printkeyans:nnn` will check for the existence of the *(sequence)*, if it does not exist it will return an error message, then it will check if not empty.

```

4945 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
4946 {
4947   \seq_if_exist:cTF { g__enumext_#3_seq }
4948   {
4949     \seq_if_empty:cF { g__enumext_#3_seq }
4950     {
4951       %%\seq_show:c { g__enumext_#3_seq }

```

If the starred argument is present we will check that the environment `enumext*` is not saved in the *(sequence)*, then execute the variable `\l__enumext_print_keyans_starred_tl` that contains the default *(keys)* for the environment `enumext*`, it will open the environment `enumext*` passing the optional argument to the “first level”, set the key `base-fix` and then will map the *(sequence)*.

```

4952       \bool_if:nTF {#1}
4953       {
4954         \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
4955         {
4956           \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
4957         }
4958       }
4959       \tl_use:N \l__enumext_print_keyans_starred_tl
4960       \begin{enumext*}[#2]
4961         \keys_set:nn { enumext / level-1 }{ base-fix }
4962         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4963       \end{enumext*}
4964     }
4965   }

```

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

```

4966   {
4967     \begin{enumext}[#2]
4968       \keys_set:nn { enumext / enumext* }{ base-fix }
4969       \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4970     \end{enumext}

```

```

4971         }
4972     }
4973 }
4974 {
4975     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
4976 }
4977 }

```

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

12.47 The command `\setenumext`

The command `\setenumext` will be in charge of managing the *⟨keys⟩* passed to all environments and to the `\printkeyans` command. We must take precautions with the `enumext*` environment and “first level” of the `enumext` environment so as not to capture *⟨keys⟩* that complicate us.

The function `__enumext_filter_first_level:n` will be in charge of filtering the *⟨keys⟩* passed to the environment `enumext*` and “first level” of the environment `enumext`.

```

\__enumext_filter_first_level:n
\__enumext_filter_first_level_key:n
\__enumext_filter_first_level_pair:nn
4978 \cs_new:Npn \__enumext_filter_first_level:n #1
4979 {
4980     \use:e
4981     {
4982         \keyval_parse:NNn
4983         \__enumext_filter_first_level_key:n
4984         \__enumext_filter_first_level_pair:nn {#1}
4985     }
4986 }

```

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

```

4987 \cs_new:Npn \__enumext_filter_first_level_key:n #1
4988 {
4989     \str_case:nnF {#1}
4990     {
4991         { resume } {}
4992         { resume* } {}
4993     }
4994     { , { \exp_not:n {#1} } } }
4995 }

```

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

```

4996 \cs_new:Npn \__enumext_filter_first_level_pair:nn #1#2
4997 {
4998     \str_case:nnF {#1}
4999     {
5000         { series } {}
5001         { resume } {}
5002         { save-ans } {}
5003     }
5004     { , { \exp_not:n {#1} } } = { \exp_not:n {#2} } }
5005 }

```

(End of definition for `__enumext_filter_first_level:n`, `__enumext_filter_first_level_key:n`, and `__enumext_filter_first_level_pair:nn`.)

Now define a “meta families” of *⟨keys⟩* to access from `\setenumext`.

```

5006 \keys_define:nn { enumext / meta-families }
5007 {
5008     enumext-1 .code:n =
5009     {
5010         \keys_set:ne { enumext / level-1 }
5011         {
5012             \__enumext_filter_first_level:n {#1}
5013         }
5014     } ,
5015     enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
5016     enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
5017     enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
5018     keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} } ,
5019     enumext* .code:n =
5020     {
5021         \keys_set:ne { enumext / enumext* }

```



```

5022         {
5023             \__enumext_filter_first_level:n {#1}
5024         }
5025     } ,
5026     keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
5027     print* .code:n = { \keys_set:nn { enumext / print } { print* = {#1} } } ,
5028     print-1 .code:n = { \keys_set:nn { enumext / print } { print-1 = {#1} } } ,
5029     print-2 .code:n = { \keys_set:nn { enumext / print } { print-2 = {#1} } } ,
5030     print-3 .code:n = { \keys_set:nn { enumext / print } { print-3 = {#1} } } ,
5031     print-4 .code:n = { \keys_set:nn { enumext / print } { print-4 = {#1} } } ,
5032     print-* .code:n = { \keys_set:nn { enumext / print } { print-* = {#1} } } ,
5033     unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
5034 }

```

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

```

5035 \seq_const_from_clist:Nn \__enumext_all_families_seq
5036 {
5037     enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
5038     keyans*, print-1, print-2, print-3, print-4, print-*, print*,
5039 }

```

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

```

5040 \NewDocumentCommand \setenumext { 0{enumext,1} +m }
5041 {
5042     \seq_clear:N \__enumext_setkey_tmpa_seq
5043     \seq_set_from_clist:Nn \__enumext_setkey_tmpb_seq {#1}
5044     \int_set:Nn \__enumext_setkey_tmpa_int
5045     {
5046         \seq_count:N \__enumext_setkey_tmpb_seq
5047     }
5048     \int_compare:nNnTF { \__enumext_setkey_tmpa_int } > { 1 }
5049     {
5050         \seq_pop_left:NN \__enumext_setkey_tmpb_seq \__enumext_setkey_tmpa_tl
5051         \seq_map_function:NN \__enumext_setkey_tmpb_seq \__enumext_set_parse:n
5052         \seq_set_map:e:NNn \__enumext_setkey_tmpa_seq \__enumext_setkey_tmpa_seq
5053         {
5054             \tl_use:N \__enumext_setkey_tmpa_tl - ##1
5055         }
5056     }
5057     {
5058         \seq_put_right:Ne \__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
5059     }
5060     \seq_if_empty:NNTF \__enumext_setkey_tmpa_seq
5061     { \seq_map_inline:Nn \__enumext_all_families_seq }
5062     { \seq_map_inline:Nn \__enumext_setkey_tmpa_seq }
5063     {
5064         \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
5065     }
5066 }

```

(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.

```

5067 \cs_new_protected:Npn \__enumext_set_parse:n #1
5068 {
5069     \tl_set:Ne \__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
5070     \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
5071     { \tl_remove_all:Nn \__enumext_setkey_tmpb_tl {##1} }
5072     \tl_if_empty:NNTF \__enumext_setkey_tmpb_tl
5073     {
5074         \seq_put_right:Ne \__enumext_setkey_tmpa_seq
5075         { \tl_trim_spaces:n {#1} }
5076     }
5077     { \__enumext_set_error:nn {#1} { } }
5078 }
5079 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
5080 { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }

```

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

12.48 The command \setenumextmeta

The command `\setenumextmeta` will be responsible for adding new “meta-keys” for the `enumext` and `enumext*` environments. The implementation code was given by Jonathan P. Spratte (@Skillmon) answer in [Add .meta key to existing keys \(l3keys\)](#).

`\setenumextmeta`

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

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

5081 \prop_const_from_keyval:Nn \c__enumext_meta_paths_prop
5082 {
5083   {enumext,1} = level-1,
5084   {enumext,2} = level-2,
5085   {enumext,3} = level-3,
5086   {enumext,4} = level-4,
5087   {enumext*} = enumext*
5088 }
```

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

```
5089 \NewDocumentCommand \setenumextmeta { s O{enumext,1} m +m }
5090 {
5091   \str_if_eq:eeTF { \tl_trim_spaces:n {#3} } { unknown }
5092   { \msg_error:nn { enumext } { prohibited-unknown } }
5093   {
5094     \bool_if:nTF {#1}
5095     {
5096       \int_step_inline:nn { 4 }
5097       { __enumext_add_meta_key:nnn { enumext, ##1 } {#3} {#4} }
5098       __enumext_add_meta_key:nnn { enumext* } {#3} {#4}
5099     }
5100     { __enumext_add_meta_key:nnn {#2} {#3} {#4} }
5101   }
5102 }
```

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

```
5103 \cs_new_protected:Npn __enumext_add_meta_key:nnn #1
5104 {
5105   \tl_set:Nn \l__enumext_meta_path_tl {#1}
5106   \tl_replace_all:Nnn \l__enumext_meta_path_tl { ~ } {}
5107   \prop_get:NVNTF
5108   \c__enumext_meta_paths_prop \l__enumext_meta_path_tl \l__enumext_meta_path_tl
5109   { __enumext_def_meta_key:Vnn \l__enumext_meta_path_tl }
5110   {
5111     \msg_error:nnn { enumext } { unknown-set } {#1}
5112     \use_none:nn
5113   }
5114 }
5115 \cs_new_protected:Npn __enumext_def_meta_key:nnn #1#2#3
5116 {
5117   \bool_lazy_or:nnTF
5118   { \keys_if_exist_p:nn { enumext / #1 } {#2} }
5119   { \keys_if_exist_p:nn { enumext / enumext* } {#2} }
5120   { \msg_error:nnn { enumext } { already-defined } {#2} }
5121   {
5122     \keys_define:nn { enumext / #1 }
5123     {
5124       #2 .meta:n = {#3},
5125       #2 .value_forbidden:n = true
5126     }
5127   }
5128 }
5129 \cs_generate_variant:Nn __enumext_def_meta_key:nnn { V }
```

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

12.49 The command \foreachkeyans

The command `\foreachkeyans` will execute a *loop* over the `<prop list>` and return its contents. The implementation code is adapted from the answer provided by Enrico Gregorio (@egreg) in [Expand a .cs defined by key inside the function](#).

`\foreachkeyans`

We define a set of `<keys>` for command and we will save the default values of these in `\g__enumext_foreach_default_keys_tl` to avoid the use of group.

```
\enumext_parse_foreach_keys:nn
__enumext_parse_foreach_keys:nn
5130 \keys_define:nn { enumext / foreach }
```

©2024 by Pablo González L

```

5131 {
5132   before .tl_set:N = \l__enumext_foreach_before_tl,
5133   before .value_required:n = true,
5134   after  .tl_set:N = \l__enumext_foreach_after_tl,
5135   after  .value_required:n = true,
5136   start  .int_set:N = \l__enumext_foreach_start_int,
5137   start  .value_required:n = true,
5138   stop   .int_set:N = \l__enumext_foreach_stop_int,
5139   stop   .value_required:n = true,
5140   step   .int_set:N = \l__enumext_foreach_step_int,
5141   step   .value_required:n = true,
5142   wrapper .cs_set_protected:Np = \__enumext_foreach_wrapper:n #1,
5143   wrapper .value_required:n = true,
5144   sep     .tl_set:N = \l__enumext_foreach_sep_tl,
5145   sep     .value_required:n = true,
5146   unknown .code:n = { \__enumext_parse_foreach_keys:n {#1} }
5147 }
5148 \keys_precompile:nnN { enumext / foreach }
5149 {
5150   before={},after={},start=1,step=1,stop=0,wrapper=#1,sep=
5151 }
5152 \g__enumext_foreach_default_keys_tl

```

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

```

5153 \cs_new_protected:Npn \__enumext_parse_foreach_keys:nn #1#2
5154 {
5155   \tl_if_blank:nTF {#2}
5156   {
5157     \msg_error:nnn { enumext } { for-key-unknown } {#1}
5158   }
5159   {
5160     \msg_error:nnnn { enumext } { for-key-value-unknown } {#1} {#2}
5161   }
5162 }
5163 \cs_new_protected:Npn \__enumext_parse_foreach_keys:n #1
5164 {
5165   \exp_args:NV \__enumext_parse_foreach_keys:nn \l_keys_key_str {#1}
5166 }

```

We create the command.

```

5167 \NewDocumentCommand \foreachkeyans { +0{ } m }
5168 {
5169   \__enumext_foreach_keyans:nn {#1} {#2}
5170 }

```

Finally the internal functions $\backslash_enumext_foreach_keyans:nn$ and $\backslash_enumext_foreach_add_body:n$ will loop through the prop list and print the contents.

```

5171 \cs_new_protected:Npn \__enumext_foreach_keyans:nn #1 #2
5172 {
5173   \tl_use:N \g__enumext_foreach_default_keys_tl
5174   \keys_set:nn { enumext / foreach } {#1}
5175   \tl_set:Nn \l__enumext_foreach_name_prop_tl {#2}
5176   \prop_if_exist:cF { g__enumext_#2_prop }
5177   {
5178     \msg_error:nnn { enumext } { undefined-storage-anskey } {#2}
5179   }
5180   \int_compare:nNnT { \l__enumext_foreach_stop_int } = { 0 }
5181   {
5182     \int_set:Nn \l__enumext_foreach_stop_int
5183     { \prop_count:c { g__enumext_#2_prop } }
5184   }
5185   \seq_clear:N \l__enumext_foreach_print_seq
5186   \int_step_function:nnnN
5187   { \l__enumext_foreach_start_int }
5188   { \l__enumext_foreach_step_int }
5189   { \l__enumext_foreach_stop_int }
5190   \__enumext_foreach_add_body:n
5191   \seq_use:NV \l__enumext_foreach_print_seq \l__enumext_foreach_sep_tl
5192 }
5193 \cs_new_protected:Npn \__enumext_foreach_add_body:n #1
5194 {
5195   \seq_put_right:Ne \l__enumext_foreach_print_seq

```

```

5196     {
5197         \exp_not:V \l__enumext_foreach_before_tl
5198         \__enumext_foreach_wrapper:n
5199         {
5200             \prop_item:cn { g__enumext_ \l__enumext_foreach_name_prop_tl _prop }{#1}
5201         }
5202         \exp_not:V \l__enumext_foreach_after_tl
5203     }
5204 }

```

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

12.50 Messages

Message used by package-load for `multicol` and `hyperref` packages.

```

5205 \msg_new:nnn { enumext } { package-load }
5206 {
5207     The ~ '#1' ~ package ~ is ~ already ~ loaded.
5208 }
5209 \msg_new:nnn { enumext } { package-not-load }
5210 {
5211     The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
5212 }
5213 \msg_new:nnn { enumext } { package-load-foot }
5214 {
5215     The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
5216 }

```

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

```

5217 \msg_new:nnn { enumext } { counters }
5218 {
5219     The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
5220     package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
5221 }

```

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

```

5222 \msg_new:nnn { enumext } { unknown-choice }
5223 {
5224     The ~ value ~ '#3' ~ for ~ '#1' ~ key ~ is ~ invalid ~ use ~ ('#2').
5225 }

```

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

```

5226 \msg_new:nnnn { enumext } { anskey-env-error }
5227 {
5228     The ~ '#1' ~ environment ~is ~ reserved ~ by ~ \\
5229     'enumext' ~ package, ~ It~ is~ already~ defined.
5230 }
5231 {
5232     The ~ anskey* ~ environment ~ is ~ defined ~ internally ~
5233     for ~ the ~ 'save-ans' ~ key.\\
5234 }

```

Message used in the creation of `⟨prop list⟩` by `enumext` package.

```

5235 \msg_new:nnn { enumext } { store-prop }
5236 {
5237     * ~ Package ~ enumext: ~ Creating ~
5238     \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
5239 }
5240 \msg_new:nnn { enumext } { store-seq }
5241 {
5242     * ~ Package ~ enumext: ~ Creating ~
5243     \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
5244 }
5245 \msg_new:nnn { enumext } { store-int }
5246 {
5247     * ~ Package ~ enumext: ~ Creating ~
5248     \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_context:.
5249 }
5250 \msg_new:nnn { enumext } { prop-seq-int-hook }
5251 {
5252     * ~ Package ~ enumext: ~ Elements ~ in ~
5253     \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
5254     * ~ Package ~ enumext: ~ Elements ~ in ~

```

```

5255     \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
5256     * ~ Package ~ enumext: ~ Value ~ off ~
5257     \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
5258 }
5259 \msg_new:nnn { enumext } { item-answer-hook }
5260 {
5261     * ~ Package ~ enumext: ~ Value ~ off ~
5262     \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\\
5263     * ~ Package ~ enumext: ~ Value ~ off ~
5264     \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\\
5265     * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
5266 }

```

Message used by [`<key = val>`] system and `\setenumext` command.

```

5267 \msg_new:nnn { enumext } { invalid-key }
5268 {
5269     The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
5270 }
5271 \msg_new:nnn { enumext } { unknown-key-family }
5272 {
5273     Unknown~key~family~`\l_keys_key_str'~for~enumext.
5274 }

```

Messages used in length calculation.

```

5275 \msg_new:nnn { enumext } { width-negative }
5276 {
5277     Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
5278     The ~ key ~ '#1'~ accepts ~ values ~ >= ~ 0pt.
5279 }
5280 \msg_new:nnn { enumext } { width-zero }
5281 {
5282     Invalid ~ '#1=#2' ~ \msg_line_context:.\
5283     The ~ key ~ '#1'~ accepts ~ values ~ > ~ 0pt.
5284 }

```

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

```

5285 \msg_new:nnn { enumext } { list-lengths }
5286 {
5287     **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
5288     \__enumext_show_length:nnn { dim } { labelsep } { #1}
5289     \__enumext_show_length:nnn { dim } { labelwidth } { #1}
5290     \__enumext_show_length:nnn { dim } { itemindent } { #1}
5291     \__enumext_show_length:nnn { dim } { leftmargin } { #1}
5292     \__enumext_show_length:nnn { dim } { rightmargin } { #1}
5293     \__enumext_show_length:nnn { dim } { listparindent } { #1}
5294     \__enumext_show_length:nnn { skip } { topsep } { #1}
5295     \__enumext_show_length:nnn { skip } { parsep } { #1}
5296     \__enumext_show_length:nnn { skip } { partopsep } { #1}
5297     \__enumext_show_length:nnn { skip } { itemsep } { #1}
5298     ****~
5299 }

```

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

```

5300 \msg_new:nnn { enumext } { list-lengths-not-nested }
5301 {
5302     **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
5303     \__enumext_show_length:nnn { dim } { labelsep } { #1}
5304     \__enumext_show_length:nnn { dim } { labelwidth } { #1}
5305     \__enumext_show_length:nnn { dim } { itemindent } { #1}
5306     \__enumext_show_length:nnn { dim } { leftmargin } { #1}
5307     \__enumext_show_length:nnn { dim } { rightmargin } { #1}
5308     \__enumext_show_length:nnn { dim } { listparindent } { #1}
5309     \__enumext_show_length:nnn { skip } { topsep } { #1}
5310     \__enumext_show_length:nnn { skip } { parsep } { #1}
5311     \__enumext_show_length:nnn { skip } { partopsep } { #1}
5312     \__enumext_show_length:nnn { skip } { itemsep } { #1}
5313     ****~
5314 }

```

Messages used by `ref` key.

```

5315 \msg_new:nnn { enumext } { key-ref-empty }
5316 {
5317     Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
5318 }

```

Messages used by `save-ans` key.

```

5319 \msg_new:nnn { enumext } { save-ans-empty }
5320 {
5321   Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1' ~ \msg_line_context:.
5322 }
5323 \msg_new:nnn { enumext } { save-ans-log }
5324 {
5325   * ~ Package ~ enumext: ~ Start ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5326 }
5327 \msg_new:nnn { enumext } { save-ans-log-hook }
5328 {
5329   * ~ Package ~ enumext: ~ Stop ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5330 }
5331 \msg_new:nnn { enumext } { save-ans-hook }
5332 {
5333   Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
5334 }
```

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

```

5335 \msg_new:nnn { enumext } { need-save-ans }
5336 {
5337   Key ~ '#1' ~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2' ~ \msg_line_context:.
5338 }
5339 \msg_new:nnn { enumext } { items-same-answer }
5340 {
5341   *****\\
5342   * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~
5343   for ~ \c_left_brace_str #2 \c_right_brace_str\\
5344   * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~
5345   'OK', ~ all ~ items ~ with ~ answer.\\
5346   *****
5347 }
5348 \msg_new:nnn { enumext } { item-greater-answer }
5349 {
5350   Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
5351   started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
5352   Items ~ > ~ Answers.
5353 }
5354 \msg_new:nnn { enumext } { item-less-answer }
5355 {
5356   Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
5357   started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
5358   Items ~ < ~ Answers.
5359 }
```

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

```

5360 \msg_new:nnn { enumext } { missing-starred }
5361 {
5362   Missing ~ '\c_backslash_str #1*' ~ #2.
5363 }
5364 \msg_new:nnn { enumext } { many-starred }
5365 {
5366   Many ~ '\c_backslash_str #1*' ~ #2.
5367 }
```

Messages used by `\printkeyans*` command.

```

5368 \msg_new:nnn { enumext } { print-starred }
5369 {
5370   \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
5371   #2 ~ environment ~ \msg_line_context:.
5372 }
```

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

```

5373 \msg_new:nnn { enumext } { list-too-deep }
5374 {
5375   Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \\
5376   The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
5377 }
```

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

```

5378 \msg_new:nnn { enumext } { anskey-unnumber-item }
5379 {
5380   Can't ~ store ~ with ~ a ~ unnumbered ~ \c_backslash_str item ~ \msg_line_context:.
```

```

5381 }
5382 \msg_new:nnn { enumext } { anskey-already-stored }
5383 {
5384   Content ~ already ~ stored ~ for ~ this ~ \c_backslash_str item ~ \msg_line_context:.
5385 }
5386 \msg_new:nnn { enumext } { anskey-empty-arg }
5387 {
5388   Can't ~ store ~ empty ~ content ~ \msg_line_context:.
5389 }
5390 \msg_new:nnn { enumext } { anskey-wrong-place }
5391 {
5392   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5393   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5394 }
5395 \msg_new:nnn { enumext } { anskey-nested }
5396 {
5397   The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
5398 }
5399 \msg_new:nnn { enumext } { anskey-math-mode }
5400 {
5401   #1 ~ can't ~ work ~ in ~ math ~ mode ~ \msg_line_context:.
5402 }
5403 \msg_new:nnn { enumext } { anskey-env-wrong }
5404 {
5405   The ~ environment ~ anskey* ~ cannot ~ use ~ in ~ '#1' ~ \msg_line_context:.
5406 }
5407 \msg_new:nnn { enumext } { ansPIC-wrong-place }
5408 {
5409   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5410   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5411 }
5412 \msg_new:nnn { enumext } { command-wrong-place }
5413 {
5414   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5415   '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
5416 }
5417 \msg_new:nnnn { enumext } { anskey-env-key-unknown }
5418 {
5419   The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5420   'anskey*' ~ and ~ is ~ being ~ ignored.
5421 }
5422 {
5423   The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5424   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5425 }
5426 \msg_new:nnnn { enumext } { anskey-env-key-value-unknown }
5427 {
5428   The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5429   'anskey*' ~ and ~ is ~ being ~ ignored.
5430 }
5431 {
5432   The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5433   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5434 }
5435 \msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
5436 { The ~ key ~ '#1' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.}
5437 {
5438   The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5439   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5440 }
5441 \msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
5442 { The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.}
5443 {
5444   The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5445   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5446 }

```

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

```

5447 \msg_new:nnn { enumext } { keyans-nested }
5448 {
5449   The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
5450 }

```



```

5451 \msg_new:nnn { enumext } { keyans-wrong-level }
5452 {
5453   Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
5454   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5455 }
5456 \msg_new:nnn { enumext } { wrong-place }
5457 {
5458   Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:~ \\
5459   '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext'.
5460 }
5461 \msg_new:nnn { enumext } { keyanspic-nested }
5462 {
5463   The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested~ \msg_line_context:~.
5464 }
5465 \msg_new:nnn { enumext } { keyanspic-wrong-level }
5466 {
5467   Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
5468   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5469 }
5470 \msg_new:nnn { enumext } { keyanspic-item-cmd }
5471 {
5472   Can't ~ use ~ \c_backslash_str item ~ in ~ keyanspic ~ \msg_line_context:.
5473 }
5474 \msg_new:nnnn { enumext } { keyans-unknown-key }
5475 {
5476   The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5477   '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5478 }
5479 {
5480   The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5481   ~ have ~ a ~ key ~ called ~ '#1'.\\
5482   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5483 }
5484 \msg_new:nnnn { enumext } { keyans-unknown-key-value }
5485 {
5486   The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5487   '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5488 }
5489 {
5490   The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5491   ~ have ~ a ~ key ~ called ~ '#1'.\\
5492   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5493 }

```

Message used by unknown $\langle keys \rangle$ in `enumext*`. environment.

```

5494 \msg_new:nnnn { enumext } { starred-unknown-key }
5495 {
5496   The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5497   '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5498 }
5499 {
5500   The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5501   ~ have ~ a ~ key ~ called ~ '#1'.\\
5502   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5503 }
5504 \msg_new:nnnn { enumext } { starred-unknown-key-value }
5505 {
5506   The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5507   '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5508 }
5509 {
5510   The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5511   ~ have ~ a ~ key ~ called ~ '#1'.\\
5512   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5513 }

```

Message used by unknown $\langle keys \rangle$ in `enumext` environment.

```

5514 \msg_new:nnnn { enumext } { standar-unknown-key }
5515 {
5516   The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_tl
5517   ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5518 }

```

```

5519 {
5520     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5521     ~ have ~ a ~ key ~ called ~ '#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5522     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5523 }
5524 \msg_new:nnnn { enumext } { standar-unknown-key-value }
5525 {
5526     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_
5527     ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5528 }
5529 {
5530     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5531     ~ have ~ a ~ key ~ called ~ '#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5532     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5533 }

```

Message used by unknown *(keys)* in `\foreachkeyans`.

```

5534 \msg_new:nnnn { enumext } { for-key-unknown }
5535 { The~key~'#1'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored.}
5536 {
5537     The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5538     Check~that~you~have~spelled~the~key~name~correctly.
5539 }
5540 \msg_new:nnnn { enumext } { for-key-value-unknown }
5541 { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored. }
5542 {
5543     The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5544     Check~that~you~have~spelled~the~key~name~correctly.
5545 }

```

Messages used by `\getkeyans` command.

```

5546 \msg_new:nnn { enumext } { undefined-storage-anskey }
5547 {
5548     Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
5549 }

```

Messages used by `\miniright` command.

```

5550 \msg_new:nnn { enumext } { missing-miniright }
5551 {
5552     Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\\
5553     The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
5554 }
5555 \msg_new:nnn { enumext } { wrong-miniright-place }
5556 {
5557     Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
5558     Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
5559 }
5560 \msg_new:nnn { enumext } { wrong-miniright-use }
5561 {
5562     Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
5563     '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
5564 }
5565 \msg_new:nnn { enumext } { wrong-miniright-starred }
5566 {
5567     Can't ~ use ~ \c_backslash_str miniright ~ in ~ starred ~ environments ~ \msg_line_context:.
5568 }
5569 \msg_new:nnn { enumext } { many-miniright-used }
5570 {
5571     Can't ~ use ~ \c_backslash_str miniright ~ more ~ than ~ once ~ \msg_line_context:.
5572 }

```

Messages used by `\setenumextmeta` command.

```

5573 \msg_new:nnn { enumext } { unknown-set }
5574 {
5575     Argument ~ [#1] ~ is ~ unknown ~ by ~ \c_backslash_str setenumextmeta ~ \msg_line_context:.
5576 }
5577 \msg_new:nnn { enumext } { already-defined }
5578 {
5579     The ~ key ~ '#1' ~ is ~ already ~ defined ~ \msg_line_context:.
5580 }
5581 \msg_new:nnn { enumext } { prohibited-unknown }
5582 {
5583     The ~ name ~ 'unknown' ~ can't ~ be ~ chosen~ for ~ a ~ meta ~ key ~ \msg_line_context:.

```

```
5584 }
```

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

```
5585 \msg_new:nnn { enumext } { nested }
```

```
5586 {
```

```
5587   The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ \msg_line_con
```

```
5588 }
```

```
5589 \msg_new:nnn { enumext } { nested-horizontal }
```

```
5590 {
```

```
5591   The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ in ~ '#1' ~ \
```

```
5592 }
```

```
5593 \msg_new:nnn { enumext } { item-joined }
```

```
5594 {
```

```
5595   Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~\msg_line_context:.
```

```
5596 }
```

```
5597 \msg_new:nnn { enumext } { item-joined-columns }
```

```
5598 {
```

```
5599   Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.
```

```
5600 }
```

12.51 Finish package

Finish package implementation.

```
5601 \file_input_stop:
```

```
5602 \</package>
```

13 Index of Implementation

The *italic* numbers denote the pages where the corresponding entry is described, the numbers underlined and all others indicate the line on which they are implemented in the package code.

Symbols	
<code>*</code>	225
<code>\+</code>	217
<code>\-</code>	217
<code>\\</code>	233, 2762, 4036, 5219, 5228, 5233, 5253, 5255, 5262, 5264, 5277, 5282, 5287, 5302, 5341, 5343, 5345, 5350, 5351, 5356, 5357, 5375, 5392, 5409, 5414, 5423, 5432, 5438, 5444, 5453, 5458, 5467, 5481, 5491, 5501, 5511, 5521, 5531, 5537, 5543, 5552, 5557, 5562
A	
<code>above</code>	<u>1577</u>
<code>above*</code>	<u>1577</u>
<code>\addvspace</code>	1146, 1175, 1218, 1221, 1389, 1392, 1489, 1495, 1530, 1536, 1557, 1563, 3573, 3708, 3723, 3919, 4265, 4279, 4320, 4334
<code>after</code>	<u>984</u>
<code>align</code>	<u>533</u>
<code>\Alpha</code>	36, <u>41</u>
<code>\Alph</code>	485, 600, 645, 713, 4928
<code>\alph</code>	36, <u>41</u>
<code>\alph</code>	486, 598, 4920
<code>\anskey</code>	12, 74, 76, <u>2580</u>
<code>anskey*</code>	13, <u>2690</u>
<code>\anspic</code>	15, 102, 105, <u>3951</u>
<code>\anspic*</code>	68
<code>\arabic</code>	30, 36
<code>\arabic</code>	484, 597, 644, 4912, 4916, 4932
B	
<code>base-fix</code>	<u>843</u>
<code>\baselineskip</code>	<u>50</u>
<code>\baselineskip</code>	860, 871
<code>before</code>	<u>984</u>
<code>before*</code>	<u>984</u>
<code>below</code>	<u>1577</u>
<code>below*</code>	<u>1577</u>
bool commands:	
<code>\bool_gset_false:N</code>	356, 357, 358, 2866, 2868, 4281, 4285, 4336
<code>\bool_gset_true:N</code>	264, 274, 1087, 2070, 2076, 4257, 4282, 4312, 4337
<code>\bool_if:NTF</code>	424, 436, 453, 1511, 1599, 1613, 1626, 1637, 1648, 1659, 1670, 1681, 1730, 1747, 1752, 1760, 1787, 1825, 1830, 1837, 1841, 1863, 1868, 1876, 1883, 1914, 1922, 2015, 2213, 2223, 2302, 2326, 2333, 2357, 2455, 2477, 2517, 2530, 2534, 2584, 2603, 2627, 2681, 2692, 2781, 2818, 2882, 2915, 2930, 3005, 3016, 3020, 3039, 3052, 3094, 3128, 3163, 3297, 3359, 3369, 3401, 3406, 3509, 3556, 3578, 3637, 3692, 3713, 3943, 3953, 4011, 4253, 4262, 4266, 4308, 4317, 4321, 4415, 4425, 4508, 4515, 4528, 4555, 4578, 4635, 4645, 4752, 4756, 4780, 4793, 4797, 4801, 4821, 4848
<code>\bool_if:nTF</code>	1537, 1564, 3150, 3279, 3317, 3846, 3986, 4952, 5094
<code>\bool_if_p:N</code>	283, 298, 856, 857, 867, 868, 1894, 1895, 1903, 1904, 2028, 2054, 2067, 2068, 2073, 2074, 2390, 2400, 2412, 2427, 2428, 2462, 2503, 2504, 2804, 2992, 2993, 3030, 3031, 3482, 3484, 3495, 3993, 3994
<code>\bool_lazy_all:nTF</code>	281, 296, 2026, 2052, 2388, 2397, 2410, 2425, 3480, 3493
<code>\bool_lazy_and:nnTF</code>	260, 270, 855, 866, 1504, 1893, 1902, 2066, 2072, 2461, 2468, 2502, 2645, 2657, 2803, 2809, 2991
<code>\bool_lazy_or:nnTF</code>	1955, 1962, 3029, 3992, 5117
<code>\bool_new:N</code>	34, 35, 36, 37, 38, 39, 40, 41, 64, 73, 95, 100, 101, 106, 107, 110, 135, 136, 144, 150, 151, 156, 158, 159, 173, 185, 187
<code>\bool_not_p:n</code>	261, 271, 2399, 2463, 2469, 2805, 2810, 3483, 3496
<code>\bool_set_eq:NN</code>	3103, 3259, 4462, 4705
<code>\bool_set_false:N</code>	433, 877, 2000, 2001, 2033, 2038, 2042, 2046, 2059, 2745, 3457, 3595, 3645, 3728, 3864, 3920, 4383, 4410, 4459, 4651, 4702
<code>\bool_set_true:N</code>	288, 289, 303, 304, 415, 419, 526, 892, 1583, 1588, 1850, 1972, 1973, 2245, 2253, 2746, 3097, 3099, 3131, 3133, 3255, 3267, 3381, 3456, 3489, 3502, 3528, 3642, 3669, 3848, 4242, 4297, 4382, 4466, 4473, 4474, 4511, 4649, 4709, 4716, 4717
box commands:	
<code>\box_dp:N</code>	1435, 1436, 1439, 1446, 1459, 1467, 1473, 1481, 3876, 3919
<code>\box_ht:N</code>	1218, 1221, 1232, 1233, 1244, 1246, 1261, 1264, 1272, 1273, 1284, 1286, 1301, 1304, 1311, 1312, 1323, 1325, 1340, 1343, 1389, 1392, 1400, 1401, 1409, 1410, 1422, 1424
<code>\box_ht_plus_dp:N</code>	3948, 3975, 3980, 4022
<code>\box_new:N</code>	70, 146, 147, 180, 186
<code>\box_use_drop:N</code>	4277, 4332, 4575, 4847
<code>\box_wd:N</code>	492
C	
<code>\c</code>	225, 226, 750, 752, 764, 766
<code>\catcode</code>	2762
<code>\cB</code>	226
<code>\cE</code>	226
<code>\centering</code>	1539, 1566, 3938, 4270, 4325
<code>check-ans</code>	<u>1992</u>
Document class:	
<code>article</code>	43
clist commands:	
<code>\clist_const:Nn</code>	192
<code>\clist_map_function:nN</code>	3924
<code>\clist_map_inline:Nn</code>	532, 798, 983, 998, 1079, 1593
<code>\clist_map_inline:nn</code>	49, 60, 78, 85, 97, 109, 138, 167, 191, 560, 580, 852, 897, 918, 1093, 1699, 1939, 2006, 2192, 2210, 2242, 2385, 2924, 3184, 3196, 3236, 3346, 3349, 3376, 3388, 3391, 3411, 5070
<code>\columnbreak</code>	75
<code>\columnbreak</code>	2465
<code>columns</code>	<u>1063</u>
<code>columns-sep</code>	<u>1063</u>
<code>\columnsep</code>	96
<code>\columnsep</code>	3551, 3690
<code>\columnseprule</code>	96
<code>\columnseprule</code>	3554, 3691
Commands provide by enumext :	
<code>\anskey</code>	28, 65, 70–74, 76, 77, 83, 85, 95, 113, 123, 124, 131
<code>\anspic*</code>	28, 29, 68, 71, 83, 84, 104, 105, 123, 124

\anspic	28, 72, 102–105, 131
\foreachkeyans	127, 134
\getkeyans	71, 123, 134
\item*	28, 29, 68, 71, 72, 83, 84, 87, 90, 114, 115, 120, 123, 124
\item	87, 90, 108, 113, 114, 116, 119
\miniright	27, 47, 55, 56, 96, 97, 134
\printkeyans*	123
\printkeyans	28, 72, 123, 124
\setenumextmeta	127, 134
\setenumext	28, 124–126, 130
Counters defined by enumext :	
enumXiii	26, 36
enumXii	26, 36
enumXiv	26, 36
enumXi	26, 36
enumXviii	26, 36
enumXvii	26, 36, 115
enumXvi	26, 36
enumXv	26, 36
cs commands:	
\cs_generate_variant:Nn	197, 198, 494, 510, 756, 772, 2294, 2299, 2375, 2698, 3336, 3926, 5129
\cs_if_exist:NTF	464
\cs_if_free:NTF	2649, 2661
\cs_new:Nn	211
\cs_new:Npn	229, 1700, 1709, 1717, 2257, 2266, 2274, 4978, 4987, 4996
\cs_new_eq:NN	383, 384, 389, 390, 438, 439, 442, 443
\cs_new_protected:Nn	221, 253, 279, 312, 342, 348, 354, 360, 366, 374, 392, 410, 621, 684, 736, 853, 999, 1003, 1007, 1011, 1015, 1019, 1023, 1027, 1031, 1035, 1039, 1043, 1047, 1051, 1055, 1059, 1094, 1106, 1130, 1148, 1159, 1177, 1203, 1224, 1349, 1375, 1395, 1428, 1450, 1485, 1491, 1594, 1608, 1622, 1633, 1644, 1655, 1666, 1677, 1758, 1861, 1874, 1891, 1912, 1940, 1945, 1970, 2011, 2021, 2064, 2079, 2086, 2095, 2100, 2105, 2110, 2119, 2124, 2129, 2300, 2324, 2331, 2355, 2362, 2376, 2601, 2620, 2636, 2699, 2735, 2766, 2801, 2843, 2864, 2872, 2913, 2928, 2956, 2989, 3025, 3037, 3050, 3136, 3146, 3157, 3275, 3291, 3432, 3449, 3478, 3507, 3514, 3535, 3565, 3576, 3618, 3635, 3659, 3676, 3701, 3711, 3862, 3922, 3927, 4031, 4039, 4043, 4062, 4072, 4103, 4232, 4251, 4287, 4306, 4369, 4397, 4404, 4413, 4423, 4444, 4593, 4633, 4664, 4670, 4687, 4744, 4862
\cs_new_protected:Npn	199, 203, 207, 235, 446, 462, 479, 489, 495, 601, 646, 718, 743, 757, 1521, 1550, 1726, 1745, 1815, 1848, 1950, 2134, 2211, 2221, 2243, 2251, 2286, 2295, 2451, 2514, 2528, 2566, 2570, 2690, 2721, 2725, 2756, 2892, 2966, 3010, 3090, 3109, 3197, 3201, 3215, 3219, 3237, 3241, 3251, 3263, 3305, 3339, 3379, 3460, 3655, 3838, 3855, 3970, 3984, 4007, 4134, 4183, 4386, 4450, 4457, 4471, 4479, 4484, 4494, 4657, 4693, 4700, 4714, 4722, 4739, 4884, 4897, 4945, 5067, 5079, 5103, 5115, 5153, 5163, 5171, 5193
\cs_new_protected_nopar:Nn	3765, 3811, 3820, 3829, 4433, 4437, 4569, 4676, 4680, 4842
\cs_new_protected_nopar:Npn	3756, 3774, 4500, 4544, 4772
\cs_set:Npn	2386, 2423, 4890
\cs_set_eq:NN	4356, 4357, 4546, 4622, 4623, 4774
\cs_set_protected:Nn	923, 939, 951, 963
\cs_set_protected:Npn	45, 54, 71, 79, 92, 98, 131, 163, 171, 511, 533, 565, 581, 628, 773, 799, 843, 879, 902, 975, 984, 1063, 1080, 1577, 1688, 1931, 1992, 2151, 2193, 2229, 2378, 2917, 3173, 3189, 3229, 3337, 3377
\cs_to_str:N	481, 504
\cs_undefine:N	2638, 2639, 2640, 2641
D	
\d	217
\DeclareDocumentEnvironment	396
dim commands:	
\dim_abs:n	3310, 3315
\dim_add:Nn	3867, 4097, 4128
\dim_compare:nNnTF	925, 941, 953, 965, 1236, 1248, 1276, 1288, 1315, 1327, 1404, 1412, 1523, 1552, 3307, 3312, 3318, 3324, 3326, 3328, 3519, 3540, 3663, 3680, 3857, 4074, 4090, 4105, 4121, 4234, 4289
\dim_compare:nTF	2487, 2831, 3438, 3624
\dim_eval:n	3919, 4018
\dim_gset_eq:NN	4243, 4298
\dim_gzero:N	2870, 4284, 4339
\dim_new:N	67, 74, 75, 76, 94, 140, 148, 149, 179, 181, 182, 188
\dim_set:Nn	492, 893, 3126, 3310, 3315, 3317, 3320, 3321, 3325, 3327, 3330, 3331, 3333, 3434, 3522, 3543, 3620, 3665, 3682, 3929, 3973, 3978, 4076, 4083, 4107, 4114, 4169, 4218, 4236, 4291, 4496
\dim_set_eq:NN	588, 635, 706, 710, 3041, 3042, 3054, 3055, 3121, 3348, 3390, 3551, 3690, 4176, 4179, 4180, 4225, 4228, 4229, 4489, 4565, 4819
\dim_sub:Nn	3443, 3629, 4092, 4123
\dim_use:N	926, 934, 1524, 1534, 2365, 2368, 2373, 3141, 3143, 3440, 3445, 3520, 3525, 3526, 3531, 3541, 3545, 3546, 3548
\dim_zero:N	3382, 3554, 3691, 3868, 3869, 3870
\dim_zero_new:N	461
\c_zero_dim	928, 942, 954, 966, 1524, 1552, 2489, 2833, 3307, 3312, 3318, 3325, 3440, 3520, 3541, 3626, 3663, 3680, 3857, 4074, 4090, 4105, 4121, 4234, 4289
\dimeval	2158
E	
\end	2328, 2359, 3570, 3705, 3915, 3940, 4954, 4963, 4970
end internal commands:	
\end__enumext_mini_page	1532, 1559, 3587, 3722, 4255, 4278, 4310, 4333
\endgroup	2762
\endlist	384
\endminipage	390
enumext	5, 3412
enumext internal commands:	
\l__enumext__ref_the_count_tl	38
\l__enumext__resume_name_tl	61
__enumext_add_meta_key:nnn	127, 5081, 5097, 5098, 5100, 5103
__enumext_add_pre_parsep:	48, 1104, 1106, 1106
__enumext_after_args_exec:	46, 999, 1011, 3425
__enumext_after_args_exec_v:	1015, 1027, 3611
__enumext_after_args_exec_vii:	1031, 1055
__enumext_after_args_exec_viii:	1059
__enumext_after_env:nn	80, 81, 83, 98, 110, 117, 203, 203, 2776, 3598, 4260, 4315, 4609
__enumext_after_hyperref:	34, 408, 410, 410
__enumext_after_list:	97, 118, 3430, 3576, 3576
\l__enumext_after_list_args_v_tl	1029
\l__enumext_after_list_args_vii_tl	1057, 4564
\l__enumext_after_list_args_viii_tl	1061, 4817

__enumext_after_list_v: ... 3616, 3659, 3711
 __enumext_after_list_vii: 113, 4367, 4404, 4404
 __enumext_after_list_viii: .. 4631, 4670, 4670
 __enumext_after_stop_list: .. 46, 97, 999, 1007, 3592
 __enumext_after_stop_list_v: 1015, 1023, 3729
 \l__enumext_after_stop_list_v_tl 1025
 __enumext_after_stop_list_vii: .. 113, 1031, 1047, 4407
 \l__enumext_after_stop_list_vii_tl ... 1049
 __enumext_after_stop_list_viii: . 1051, 4673
 \l__enumext_after_stop_list_viii_tl ... 1053
 \l__enumext_align_label_vii_str .. 4530, 4537
 \l__enumext_align_label_viii_str . 4803, 4810
 \l__enumext_align_label_X_str 171
 \c__enumext_all_envs_clist .. 192, 532, 798, 983, 998, 1079, 1593
 \c__enumext_all_families_seq .. 126, 5035, 5061
 \l__enumext_anskey_env_bool 31, 79, 34, 289, 304, 2692
 __enumext_anskey_env_clean_vars: . 82, 2797, 2801, 2864
 __enumext_anskey_env_define_keys: 79, 2690, 2699, 2770
 __enumext_anskey_env_exec: 81, 2695, 2766, 2766
 __enumext_anskey_env_make:n 65, 79, 1975, 2690, 2690, 2698
 __enumext_anskey_env_reset_keys: 80, 81, 2735, 2798
 __enumext_anskey_env_reset_keys:__-enumext_rescan_anskey_env:n 2690
 __enumext_anskey_env_save_keys: .. 81, 2778, 2801, 2801
 __enumext_anskey_env_store: .. 82, 2794, 2801, 2843
 __enumext_anskey_env_unknown:n 80, 2718, 2721
 __enumext_anskey_env_unknown:nn . 2723, 2725
 \l__enumext_anskey_level_int .. 28, 2622, 2623
 __enumext_anskey_safe_inner: . 78, 2595, 2601, 2620
 __enumext_anskey_safe_inner:n 77
 __enumext_anskey_safe_outer: . 77, 2582, 2601, 2601
 __enumext_anskey_show_wrap_arg:n . 76, 2514, 2514, 2532, 2547
 __enumext_anskey_show_wrap_left:n 76, 2459, 2528, 2528
 __enumext_anskey_unknown:n 77, 2550, 2564, 2566
 __enumext_anskey_unknown:nn . 2550, 2568, 2570
 __enumext_anskey_wrapper:n 2155, 2526
 \l__enumext_anspic_body_box ... 139, 3972, 3975
 \l__enumext_anspic_body_htdp_dim . 139, 3973, 4021
 __enumext_anspic_box_set_dim:n .. 3970, 3970, 4010
 __enumext_anspic_label:nn ... 3984, 4013, 4027
 \l__enumext_anspic_label_box .. 139, 3977, 3980
 \l__enumext_anspic_label_htdp_dim 139, 3978, 4020
 __enumext_anspic_start_list_tag: 3783, 3811, 4033
 __enumext_anspic_stop_list_tag: . 3783, 3829, 4037
 __enumext_anspic_stop_start_list_tag: 3783, 3820, 4035
 __enumext_at_begin_document:n 33, 34, 199, 199, 381, 387
 \l__enumext_base_line_fix_bool . 847, 857, 868, 877
 __enumext_before_args_exec: .. 46, 96, 112, 999, 999, 3517
 __enumext_before_args_exec_v: 1015, 1015, 3662
 __enumext_before_args_exec_vii: . 1031, 1031, 4401
 __enumext_before_args_exec_viii: 1035, 4667
 __enumext_before_env:nn 79, 203, 207, 2643, 2655, 2667, 2768
 __enumext_before_keys_exec: 46, 999, 1003, 3422
 __enumext_before_keys_exec_v: 1015, 1019, 3608
 __enumext_before_keys_exec_vii 1031
 __enumext_before_keys_exec_vii: . 1039, 4350
 __enumext_before_keys_exec_viii: 1043, 4618
 __enumext_before_list: ... 96, 3416, 3514, 3514
 __enumext_before_list_v: ... 3603, 3659, 3659
 __enumext_before_list_vii: ... 112, 4345, 4397, 4397
 __enumext_before_list_viii: .. 118, 4614, 4664, 4664
 \l__enumext_before_no_starred_key_v_tl 1021
 \l__enumext_before_no_starred_key_vii_tl 1041
 \l__enumext_before_no_starred_key_viii_tl 1045
 \l__enumext_before_starred_key_v_tl ... 1017
 \l__enumext_before_starred_key_vii_tl . 1033
 \l__enumext_before_starred_key_viii_tl 1037
 __enumext_calc_hspace:NNNNNNN 91, 3305, 3305, 3336, 3341, 3383
 __enumext_check_ans_active: . 66, 96, 112, 2011, 2011, 3518, 4400
 \g__enumext_check_ans_item_tl 85
 \g__enumext_check_ans_key_bool 67, 68, 150, 356, 2070, 2076, 2882
 \l__enumext_check_ans_key_bool 67, 1996, 2001, 2067, 2073
 __enumext_check_ans_key_hook: 67, 97, 113, 2064, 2064, 3593, 4408
 __enumext_check_ans_level: 66, 2011, 2017, 2021
 __enumext_check_ans_log: 67, 68, 83, 2110, 2110, 2886
 __enumext_check_ans_log_msg_greater: 2110, 2116, 2129
 __enumext_check_ans_log_msg_less: 2110, 2114, 2119
 __enumext_check_ans_log_msg_same_ok: 2110, 2115, 2124
 __enumext_check_ans_msg_greater: 2086, 2092, 2105
 __enumext_check_ans_msg_less: 2086, 2090, 2095
 __enumext_check_ans_msg_same_ok: 2086, 2091, 2100
 __enumext_check_ans_show: .. 67, 82, 2086, 2086, 2884
 \l__enumext_check_answers_bool . 65, 66, 77, 87, 150, 1973, 2000, 2015, 2302, 2326, 2333, 2357, 2584, 2781, 3005, 3094, 3128, 4508
 __enumext_check_starred_cmd:n 32, 68, 85, 117, 2134, 2134, 3614, 3917, 4629

`\g__enumext_check_starred_cmd_int` [150](#), [2137](#),
[2143](#), [2148](#), [3273](#), [3991](#), [4751](#)
`\l__enumext_check_start_line_env_tl` . [32](#), [150](#),
[319](#), [327](#), [335](#), [2140](#), [2146](#), [2149](#)
`\l__enumext_columns_sep_v_dim` [3680](#), [3682](#), [3690](#)
`\l__enumext_columns_sep_vii_dim` . [4074](#), [4076](#),
[4085](#), [4097](#), [4173](#), [4590](#)
`\l__enumext_columns_sep_viii_dim` . [4105](#), [4107](#),
[4116](#), [4128](#), [4222](#), [4859](#)
`\l__enumext_columns_v_int` [1369](#), [1387](#), [1555](#), [3678](#),
[3686](#), [3698](#), [3703](#)
`\l__enumext_columns_vii_int` . [4079](#), [4082](#), [4086](#),
[4095](#), [4137](#), [4141](#), [4144](#), [4150](#), [4156](#), [4160](#), [4584](#), [4598](#)
`\l__enumext_columns_viii_int` . [4110](#), [4113](#), [4117](#),
[4126](#), [4186](#), [4190](#), [4193](#), [4199](#), [4205](#), [4209](#), [4853](#), [4868](#)
`\l__enumext_counter_i_tl` [45](#), [471](#)
`\l__enumext_counter_ii_tl` [45](#), [472](#)
`\l__enumext_counter_iii_tl` [45](#), [473](#)
`\l__enumext_counter_iv_tl` [45](#), [474](#)
`\c__enumext_counter_style_tl` [30](#), [50](#), [223](#)
`\g__enumext_counter_styles_tl` . [26](#), [36](#), [67](#), [482](#),
[500](#)
`\l__enumext_counter_v_tl` [45](#), [475](#), [726](#)
`\l__enumext_counter_vi_tl` [45](#), [476](#)
`\l__enumext_counter_vii_tl` [45](#), [477](#), [656](#)
`\l__enumext_counter_viii_tl` [45](#), [478](#), [673](#)
`\l__enumext_current_widest_dim` [26](#), [67](#), [506](#), [589](#),
[636](#), [707](#), [711](#)
`__enumext_def_meta_key:nnn` . . . [127](#), [5081](#), [5109](#),
[5115](#), [5129](#)
`__enumext_default_item:n` . . . [3090](#), [3090](#), [3154](#)
`__enumext_define_counters:Nn` [26](#), [462](#), [462](#), [471](#),
[472](#), [473](#), [474](#), [475](#), [476](#), [477](#), [478](#)
`__enumext_endminipage:` . [34](#), [387](#), [390](#), [404](#), [4571](#),
[4844](#)
`\g__enumext_envir_name_tl` [32](#), [34](#), [290](#), [305](#), [364](#),
[1943](#), [1948](#), [1958](#), [2098](#), [2103](#), [2108](#), [2122](#), [2127](#), [2132](#)
`\l__enumext_envir_name_tl` . [31](#), [32](#), [34](#), [259](#), [269](#),
[318](#), [326](#), [334](#), [5477](#), [5480](#), [5487](#), [5490](#), [5497](#), [5500](#),
[5507](#), [5510](#), [5516](#), [5520](#), [5526](#), [5530](#), [5587](#), [5591](#)
`__enumext_execute_after_env:` [33](#), [64](#), [67](#), [68](#), [78](#),
[82](#), [2872](#), [2872](#), [3598](#), [4609](#)
`__enumext_fake_item:` [923](#), [923](#), [3368](#)
`\l__enumext_fake_item_indent_v_dim` [942](#), [947](#)
`\l__enumext_fake_item_indent_v_tl` [944](#), [3256](#),
[3260](#), [3268](#)
`\l__enumext_fake_item_indent_vii_dim` [954](#), [959](#)
`\l__enumext_fake_item_indent_vii_tl` [956](#), [4567](#)
`\l__enumext_fake_item_indent_viii_dim` . [966](#),
[971](#), [4825](#)
`\l__enumext_fake_item_indent_viii_tl` . [968](#),
[4823](#), [4828](#)
`\l__enumext_fake_item_indent_X_tl` [98](#)
`__enumext_fake_item_vii:` [923](#), [951](#), [3400](#)
`__enumext_fake_item_viii:` [923](#), [963](#), [3405](#)
`__enumext_fake_make_label_vii:n` . [115](#), [4500](#),
[4500](#), [4561](#)
`__enumext_filter_first_level:n` . . [125](#), [4978](#),
[4978](#), [5012](#), [5023](#)
`__enumext_filter_first_level_key:n` [125](#), [4978](#),
[4983](#), [4987](#)
`__enumext_filter_first_level_pair:nn` . [125](#),
[4978](#), [4984](#), [4996](#)
`__enumext_filter_save_key:n` . . [71](#), [2218](#), [2226](#),
[2249](#), [2255](#), [2257](#), [2257](#), [4910](#), [4914](#), [4918](#), [4922](#), [4926](#),
[4930](#)
`__enumext_filter_save_key_key:n` . . [71](#), [2257](#),
[2262](#), [2266](#)
`__enumext_filter_save_key_pair:nn` [71](#), [2257](#),
[2263](#), [2274](#)
`__enumext_filter_series:n` [59](#), [1700](#), [1700](#), [1738](#),
[1750](#), [1755](#)
`__enumext_filter_series_key:n` [59](#), [1700](#), [1705](#),
[1709](#)
`__enumext_filter_series_pair:nn` . . [60](#), [1700](#),
[1706](#), [1717](#)
`__enumext_first_item_tmp_vii:` [111](#), [113](#), [4356](#),
[4433](#), [4433](#)
`__enumext_first_item_tmp_viii:` [117](#), [119](#), [4622](#),
[4676](#), [4676](#)
`\g__enumext_footnote_arg_seq` . [168](#), [4045](#), [4058](#),
[4068](#)
`\g__enumext_footnote_int` . [168](#), [4052](#), [4055](#), [4057](#),
[4059](#)
`\g__enumext_footnote_int_seq` . [168](#), [4046](#), [4059](#),
[4064](#), [4067](#)
`__enumext_footnotes_key_bool` [34](#)
`\l__enumext_footnotes_key_bool` [29](#), [35](#), [116](#), [158](#),
[419](#), [424](#), [433](#), [4555](#), [4578](#), [4793](#), [4848](#)
`__enumext_footnotetext:nn` . . . [4039](#), [4039](#), [4069](#)
`__enumext_foreach_add_body:n` . [128](#), [5130](#), [5190](#),
[5193](#)
`\l__enumext_foreach_after_tl` [5134](#), [5202](#)
`\l__enumext_foreach_before_tl` [5132](#), [5197](#)
`\g__enumext_foreach_default_keys_tl` [127](#), [124](#),
[5152](#), [5173](#)
`__enumext_foreach_keyans:nn` . . [128](#), [5130](#), [5169](#),
[5171](#)
`\l__enumext_foreach_name_prop_tl` . [124](#), [5175](#),
[5200](#)
`\l__enumext_foreach_print_seq` [124](#), [5185](#), [5191](#),
[5195](#)
`\l__enumext_foreach_sep_tl` [5144](#), [5191](#)
`\l__enumext_foreach_start_int` [5136](#), [5187](#)
`\l__enumext_foreach_step_int` [5140](#), [5188](#)
`\l__enumext_foreach_stop_int` . [5138](#), [5180](#), [5182](#),
[5189](#)
`__enumext_foreach_wrapper:n` [5142](#), [5198](#)
`__enumext_getkeyans:nn` . . [123](#), [4893](#), [4897](#), [4897](#)
`__enumext_getkeyans_aux:n` [123](#), [4881](#), [4884](#), [4884](#)
`\l__enumext_hyperref_bool` . [29](#), [34](#), [35](#), [158](#), [415](#),
[436](#), [453](#), [2504](#), [2993](#), [4780](#)
`__enumext_hypertarget:nn` [35](#), [410](#), [438](#), [442](#), [458](#)
`__enumext_if_is_int:n` [215](#)
`__enumext_if_is_int:nTF` [215](#), [745](#), [759](#)
`__enumext_internal_mini_page:` [34](#), [94](#), [112](#), [392](#),
[392](#), [3451](#), [4371](#)
`__enumext_is_not_nested:` [26](#), [31](#), [94](#), [112](#), [253](#), [253](#),
[3452](#), [4372](#)
`__enumext_is_on_first_level:` . [26](#), [31](#), [95](#), [112](#),
[253](#), [279](#), [3458](#), [4384](#)
`\g__enumext_item_anskey_int` [77](#), [85](#), [150](#), [351](#), [378](#),
[379](#), [2083](#), [2453](#), [3007](#)
`__enumext_item_answer_diff:` . . [67](#), [68](#), [82](#), [2079](#),
[2079](#), [2879](#)
`\g__enumext_item_answer_diff_int` . [67](#), [68](#), [150](#),
[352](#), [2081](#), [2088](#), [2112](#)
`\l__enumext_item_column_pos_vii_int` [113](#), [4144](#),
[4150](#), [4156](#), [4160](#), [4167](#), [4440](#), [4584](#), [4587](#)


```

\l__enumext_item_column_pos_viii_int .. 119,
    4193, 4199, 4205, 4209, 4216, 4683, 4853, 4856
l__enumext_item_column_pos_X_int ..... 171
\g__enumext_item_count_all_vii_int 113, 4168,
    4441, 4598, 4606
\g__enumext_item_count_all_viii_int 119, 4217,
    4684, 4867, 4876
\g__enumext_item_count_all_X_int ..... 171
\g__enumext_item_number_bool ..... 150
\l__enumext_item_number_bool 66, 156, 2033, 2038,
    2042, 2046, 2059, 2627, 2681, 3097, 3131, 4511
\g__enumext_item_number_int 66, 67, 150, 350, 377,
    379, 2032, 2037, 2041, 2045, 2058, 2083, 3096, 3130,
    4510
\__enumext_item_peek_args_vii: 113, 114, 4442,
    4444, 4444
\__enumext_item_peek_args_viii: .. 119, 4685,
    4687, 4687
\__enumext_item_star_exec: . 88, 3109, 3136, 3165
\l__enumext_item_starred_vii_bool 4459, 4473,
    4515
\l__enumext_item_starred_viii_bool 4702, 4716,
    4797, 4821
\l__enumext_item_starred_X_bool ..... 171
\__enumext_item_std:w .. 33, 87, 90, 103, 381, 385,
    3100, 3106, 3134, 3256, 3260, 3268, 4835, 4838
\g__enumext_item_symbol_aux_tl . 87, 128, 3114,
    3117, 3142, 3170
\g__enumext_item_symbol_aux_vii_tl 4481, 4517,
    4520, 4524, 4526
\g__enumext_item_symbol_aux_X_tl ..... 171
\l__enumext_item_symbol_sep_vii_dim .. 4489,
    4496, 4523, 4525
\l__enumext_item_symbol_vii_tl ..... 4520
\l__enumext_item_text_vii_box .... 4547, 4575
\l__enumext_item_text_viii_box ... 4787, 4847
\l__enumext_item_text_X_box ..... 171
\l__enumext_item_width_vii_dim ... 4083, 4092,
    4171, 4179, 4180
\l__enumext_item_width_viii_dim .. 4114, 4123,
    4220, 4228, 4229
\l__enumext_item_width_X_dim ..... 171
\l__enumext_itemindent_X_dim ..... 71
\l__enumext_itemsep_i_skip ... 1230, 1237, 1240,
    1242, 1249, 1253, 1256, 1258, 1398, 1405, 1407, 1408,
    1413, 1417, 1419, 1420
\l__enumext_itemsep_ii_skip .. 1270, 1277, 1280,
    1282, 1289, 1293, 1296, 1298
\l__enumext_itemsep_iii_skip . 1309, 1316, 1319,
    1321, 1328, 1332, 1335, 1337
\l__enumext_itemsep_vii_skip ..... 4604
\l__enumext_itemsep_viii_skip ..... 4874
\l__enumext_joined_item_aux_vii_int .. 4165,
    4166, 4167, 4168, 4174
\l__enumext_joined_item_aux_viii_int . 4214,
    4215, 4216, 4217, 4223
\l__enumext_joined_item_aux_X_int .... 171
\__enumext_joined_item_vii:w . 114, 4447, 4448,
    4450, 4450
\l__enumext_joined_item_vii_int .. 4136, 4137,
    4140, 4142, 4148, 4153, 4158, 4163, 4165, 4171
\__enumext_joined_item_viii:w . 119, 4690, 4691,
    4693, 4693
\l__enumext_joined_item_viii_int . 4185, 4186,
    4189, 4191, 4197, 4202, 4207, 4212, 4214, 4220
\l__enumext_joined_item_X_int ..... 171
\l__enumext_joined_width_vii_dim . 4169, 4176,
    4179, 4549, 4563
\l__enumext_joined_width_viii_dim 4218, 4225,
    4228, 4789, 4818
\l__enumext_joined_width_X_dim ..... 171
\__enumext_keyans_addto_prop:n 83, 2892, 2892,
    3270, 3988
\__enumext_keyans_addto_seq:n . 84, 2966, 2966,
    3272, 3990
\__enumext_keyans_addto_seq_link: 2966, 2987,
    2989, 4750
\__enumext_keyans_anspic_code:nnn 105, 3967,
    3970, 4031
\__enumext_keyans_anspic_label:nnn 3970, 4007,
    4034
\__enumext_keyans_default_item:n .. 90, 3251,
    3251, 3287
\l__enumext_keyans_env_bool 34, 3483, 3496, 3642,
    3728
\__enumext_keyans_fake_item: .. 923, 939, 3358
\l__enumext_keyans_level_h_int .. 117, 28, 666,
    693, 2611, 2673, 2944, 4378, 4639, 4640
\l__enumext_keyans_level_int .. 28, 1515, 2607,
    2669, 2939, 3641, 3646, 3961
\__enumext_keyans_make_label: 37, 91, 3275, 3291,
    3356
\__enumext_keyans_mini_right_cmd:n 56, 1517,
    1550, 1550
\__enumext_keyans_mini_set_vskip: ..... 53
\__enumext_keyans_minipage_add_space: 1349,
    1375, 3671
\__enumext_keyans_minipage_set_skip: . 1349,
    1349, 1377
\__enumext_keyans_multi_addvspace: 1148, 1159,
    3695
\__enumext_keyans_multi_set_vskip: 49, 1148,
    1148, 1161
\__enumext_keyans_multicols_start: 3659, 3674,
    3676
\__enumext_keyans_multicols_stop: 1554, 3659,
    3701, 3726
\__enumext_keyans_name_and_start: 26, 32, 117,
    312, 312, 3643, 3845, 4644
\__enumext_keyans_parse_keys:n 3602, 3655, 3655
\l__enumext_keyans_pic_above_int . 139, 3930,
    3931, 3933
\l__enumext_keyans_pic_above_skip . 103, 139,
    3874, 3901
\__enumext_keyans_pic_arg_two: 103, 3862, 3862,
    3884
\l__enumext_keyans_pic_below_int . 139, 3930,
    3931, 3934
\l__enumext_keyans_pic_body_seq 104, 105, 139,
    3911, 3939, 3965
\__enumext_keyans_pic_do:n 104, 3911, 3913, 3922,
    3922, 3926
\l__enumext_keyans_pic_label_pos_str .. 102,
    139, 3849, 3852, 3937
\l__enumext_keyans_pic_level_int .. 28, 1499,
    2615, 2677, 2895, 2934, 2969, 3057, 3840, 3841
\__enumext_keyans_pic_row:n .. 104, 3924, 3927,
    3927

```

```

\__enumext_keyans_pic_safe_exec:n    102, 3838,
    3838, 3883
\__enumext_keyans_pic_skip_abs:N    . 103, 3855,
    3855, 3873
\l__enumext_keyans_pic_star_bool    .. 102, 139,
    3848, 3943, 4011
\l__enumext_keyans_pic_width_dim    . 139, 3929,
    3937, 4001
\__enumext_keyans_pre_itemsep_skip: .. 1349,
    1368, 1395
\__enumext_keyans_redefine_item:    .. 91, 3275,
    3275, 3355
\__enumext_keyans_ref:    .... 41, 718, 736, 3357
\__enumext_keyans_ref:n    .... 40, 715, 718, 718
\__enumext_keyans_safe_exec:    . 3601, 3635, 3635
\__enumext_keyans_set_item_width:    . 98, 3610,
    3618, 3618
\__enumext_keyans_show_ans:    .. 3010, 3018, 3037
\__enumext_keyans_show_item_opt:    . 3010, 3025,
    3268, 4004, 4824
\__enumext_keyans_show_left:n    . 90, 3010, 3010,
    3266, 3997
\__enumext_keyans_show_pos:    .. 3010, 3022, 3050
\__enumext_keyans_starred_item:n    .. 90, 3263,
    3263, 3283
\__enumext_keyans_store_ref:    .. 83, 2913, 2913,
    3271, 3989, 4748
\__enumext_keyans_store_ref_aux_i:    84, 2913,
    2925, 2928
\__enumext_keyans_store_ref_aux_ii: 84, 2913,
    2954, 2956
\__enumext_keyans_unknown_keys:n    . 3189, 3193,
    3197
\__enumext_keyans_unknown_keys:nn 3189, 3199,
    3201
\__enumext_keyans_wrapper_opt:n    .. 2161, 3033
\l__enumext_label_copy_i_tl    .. 2419, 2932, 2937,
    2942, 2947
\l__enumext_label_copy_v_tl    ..... 2942
\l__enumext_label_copy_vi_tl    ..... 2937
\l__enumext_label_copy_vii_tl    2395, 2406, 2435,
    2932
\l__enumext_label_copy_viii_tl    ..... 2947
\l__enumext_label_copy_X_tl    ..... 160
\l__enumext_label_fill_left_v_tl    .... 3295
\l__enumext_label_fill_left_X_tl    .... 98
\l__enumext_label_fill_right_v_tl    .... 3302
\l__enumext_label_fill_right_X_tl    .... 98
\l__enumext_label_font_style_v_tl    3296, 4003
\l__enumext_label_font_style_vii_tl .. 4532,
    4539
\l__enumext_label_font_style_viii_tl . 4805,
    4812
\l__enumext_label_i_tl    ..... 581
\l__enumext_label_ii_tl    ..... 581
\l__enumext_label_iii_tl    ..... 581
\l__enumext_label_iv_tl    ..... 581
\__enumext_label_style:Nnn    26, 36, 495, 495, 510,
    586, 633, 704, 708
\l__enumext_label_v_tl    .. 83, 84, 701, 2900, 2974,
    3044, 3084, 3265, 3269, 3605, 3977, 3996, 3998
\l__enumext_label_vi_tl    . 83, 84, 701, 2897, 2971,
    3996, 3998, 4004
\l__enumext_label_vii_tl    . 628, 4468, 4491, 4498
\l__enumext_label_viii_tl    628, 4711, 4742, 4746
\l__enumext_label_width_by_box    .. 67, 491, 492
\__enumext_label_width_by_box:Nn    36, 489, 489,
    494, 506, 769
\l__enumext_labelsep_i_dim    ... 3042, 3047, 3055,
    3087, 4754, 4769
\l__enumext_labelsep_v_dim    ..... 3685
\l__enumext_labelsep_vii_dim    . 2519, 3042, 3055,
    4078, 4088, 4172, 4435, 4489, 4542, 4551
\l__enumext_labelsep_viii_dim    4109, 4119, 4221,
    4678, 4791, 4816, 4825
\l__enumext_labelwidth_i_dim    . 3041, 3047, 3054,
    3087, 4754, 4769
\l__enumext_labelwidth_v_dim    ..... 3685
\l__enumext_labelwidth_vii_dim    ... 2519, 3041,
    3054, 4078, 4087, 4172, 4435, 4530, 4537, 4550
\l__enumext_labelwidth_viii_dim    .. 4109, 4118,
    4221, 4678, 4790, 4803, 4810
\l__enumext_leftmargin_tmp_v_bool    . 103, 3864
\l__enumext_leftmargin_tmp_X_bool    .... 71
\l__enumext_leftmargin_tmp_X_dim    ..... 71
\l__enumext_leftmargin_X_dim    ..... 71
\__enumext_level:    211, 211, 610, 613, 614, 623, 625,
    926, 930, 934, 1001, 1005, 1009, 1013, 1096, 1098,
    1100, 1102, 1135, 1137, 1139, 1141, 1146, 1181, 1187,
    1192, 1194, 1197, 1200, 1213, 1216, 1524, 1528, 1534,
    1597, 1599, 1601, 1604, 1611, 1613, 1615, 1618, 2213,
    2215, 2217, 2245, 2246, 2248, 2304, 2312, 2316, 2320,
    2523, 2524, 3099, 3100, 3104, 3105, 3106, 3114, 3122,
    3123, 3126, 3133, 3134, 3138, 3141, 3143, 3161, 3162,
    3163, 3166, 3169, 3419, 3421, 3440, 3445, 3489, 3502,
    3509, 3520, 3522, 3525, 3526, 3528, 3531, 3538, 3541,
    3543, 3545, 3546, 3547, 3548, 3551, 3556, 3562, 3568,
    3573, 3578
\l__enumext_level_h_int    112, 28, 262, 285, 299, 649,
    686, 1506, 2029, 2049, 2414, 2647, 2659, 3497, 4373,
    4374
\l__enumext_level_int    . 94, 28, 213, 272, 284, 300,
    394, 1108, 1226, 1505, 2023, 2055, 2391, 2401, 2407,
    2413, 2420, 2429, 2434, 2646, 2658, 2874, 3371, 3453,
    3454, 3465, 3473, 3487, 3500, 3552, 3650, 3957, 4417,
    4427, 4652, 5517, 5521, 5527, 5531
\__enumext_list_arg_two_i:    ..... 3337
\__enumext_list_arg_two_ii:    ..... 3337
\__enumext_list_arg_two_iii:    ..... 3337
\__enumext_list_arg_two_iv:    ..... 3337
\__enumext_list_arg_two_v:    . 91, 3337, 3607, 3865
\__enumext_list_arg_two_vii:    .... 3377, 4349
\__enumext_list_arg_two_viii:    .... 3377, 4617
\l__enumext_listoffset_v_dim    . 3626, 3631, 3687
\l__enumext_listparindent_vii_dim    .... 4565
\l__enumext_listparindent_viii_dim    ... 4819
\__enumext_log_answer_vars:    . 33, 366, 374, 2881
\__enumext_log_global_vars:    . 33, 366, 366, 2880
\__enumext_make_label    ..... 3146
\__enumext_make_label:    .... 37, 88, 3157, 3366
\l__enumext_mark_answer_sym_tl    73, 2167, 2370,
    2536, 3059, 3072, 4758
\l__enumext_mark_position_str    128, 2171, 2172,
    2198, 2199, 2368
\l__enumext_mark_ref_sym_tl    .. 2184, 2509, 3001
\l__enumext_meta_path_tl    . 124, 5105, 5106, 5108,
    5109
\c__enumext_meta_paths_prop    ..... 127, 5081

```

```

\__enumext_mini_addvspace_vii: 55, 1485, 1485,
    4246
\__enumext_mini_addvspace_viii: 55, 1485, 1491,
    4301
__enumext_mini_env* ..... 392
\__enumext_mini_page 1534, 1561, 3531, 3672, 4248,
    4264, 4303, 4319
\__enumext_mini_right_cmd:n 56, 1519, 1521, 1521
\__enumext_mini_set_vskip_vii: 54, 1428, 1428,
    1487
\__enumext_mini_set_vskip_viii: 54, 1428, 1450,
    1493
\__enumext_minipage:w 34, 387, 389, 398, 4563, 4818
\l__enumext_minipage_active_v_bool 3669, 3692,
    3713
\g__enumext_minipage_active_vii_bool .. 110,
    4257, 4262, 4281
\l__enumext_minipage_active_vii_bool . 4242,
    4253
\g__enumext_minipage_active_viii_bool 4312,
    4317, 4336
\l__enumext_minipage_active_viii_bool 4297,
    4308
\g__enumext_minipage_active_X_bool ... 171
\l__enumext_minipage_active_X_bool .... 86
\__enumext_minipage_add_space: .. 50, 96, 1177,
    1203, 3530
\g__enumext_minipage_after_skip 86, 1432, 1444,
    4279, 4334
\l__enumext_minipage_after_skip .. 50, 97, 86,
    1190, 1230, 1232, 1237, 1240, 1244, 1249, 1253, 1256,
    1260, 1272, 1277, 1280, 1284, 1289, 1293, 1296, 1300,
    1311, 1316, 1319, 1323, 1328, 1332, 1335, 1339, 1351,
    1365, 1398, 1400, 1405, 1407, 1409, 1413, 1417, 1419,
    1421, 1452, 1465, 1479, 1530, 1557, 3723
\g__enumext_minipage_center_vii_bool . 4266,
    4282
\g__enumext_minipage_center_viii_bool 4321,
    4337
\g__enumext_minipage_center_X_bool ... 171
\l__enumext_minipage_hsep_v_dim ..... 3667
\l__enumext_minipage_hsep_vii_dim .... 4240
\l__enumext_minipage_hsep_viii_dim ... 4295
\l__enumext_minipage_left_skip 86, 1352, 1430,
    1435, 1439, 1453, 1457, 1471, 1489, 1495
\l__enumext_minipage_left_v_dim .. 3665, 3672
\l__enumext_minipage_left_vii_dim 4236, 4248
\l__enumext_minipage_left_viii_dim 4291, 4303
\l__enumext_minipage_left_X_dim ..... 86
\g__enumext_minipage_right_skip 86, 1431, 1436,
    1440, 4265, 4320
\l__enumext_minipage_right_skip . 50, 86, 1179,
    1185, 1190, 1192, 1194, 1353, 1354, 1360, 1365, 1366,
    1367, 1372, 1454, 1461, 1475, 1536, 1563
\l__enumext_minipage_right_v_dim . 1552, 1561,
    3663, 3667
\g__enumext_minipage_right_vii_dim 109, 4244,
    4264, 4284
\l__enumext_minipage_right_vii_dim 109, 4234,
    4239, 4245
\g__enumext_minipage_right_viii_dim .. 4299,
    4319, 4339
\l__enumext_minipage_right_viii_dim .. 4289,
    4294, 4300
\g__enumext_minipage_right_X_dim ..... 171
\g__enumext_minipage_right_X_skip .... 171
\__enumext_minipage_set_skip: . 50, 1177, 1177,
    1205
\g__enumext_minipage_stat_int 96, 86, 1541, 1568,
    3529, 3580, 3585, 3670, 3715, 3720
\l__enumext_miniright_code_vii_box 4273, 4277
\g__enumext_miniright_code_vii_tl 110, 4268,
    4275, 4283
\l__enumext_miniright_code_viii_box .. 4328,
    4332
\g__enumext_miniright_code_viii_tl 4323, 4330,
    4338
\l__enumext_miniright_code_X_box ..... 171
\__enumext_multi_addvspace: . 49, 97, 1130, 1130,
    3559
\__enumext_multi_set_vskip: 48, 1094, 1094, 1132
\l__enumext_multicols_above_ii_skip ... 1113
\l__enumext_multicols_above_iii_skip .. 1119
\l__enumext_multicols_above_iv_skip ... 1125
\l__enumext_multicols_above_v_skip 1150, 1164,
    1175, 1366
\l__enumext_multicols_above_X_skip .... 79
\l__enumext_multicols_below_ii_skip .. 1233,
    1242, 1246, 1258, 1263
\l__enumext_multicols_below_iii_skip . 1273,
    1282, 1286, 1298, 1303
\l__enumext_multicols_below_iv_skip .. 1312,
    1321, 1325, 1337, 1342
\l__enumext_multicols_below_v_skip 1154, 1168,
    1367, 1401, 1408, 1410, 1420, 1423, 3708
\l__enumext_multicols_below_X_skip .... 79
\g__enumext_multicols_right_X_skip .... 79
\__enumext_multicols_start: 96, 3533, 3535, 3535
\__enumext_multicols_stop: 97, 1526, 3565, 3565,
    3590
\__enumext_nested_base_line_fix: . 43, 95, 112,
    843, 853, 3469, 4394
\__enumext_newlabel:nn 29, 35, 74, 446, 446, 2445,
    2960
\l__enumext_newlabel_arg_one_tl 29, 35, 74, 84,
    160, 2438, 2446, 2508, 2949, 2961, 2999
\l__enumext_newlabel_arg_two_tl 29, 35, 73, 160,
    2394, 2404, 2417, 2432, 2447, 2936, 2941, 2946, 2962
\__enumext_parse_foreach_keys:n .. 5130, 5146,
    5163
\__enumext_parse_foreach_keys:nn . 5130, 5153,
    5165
\__enumext_parse_keys:n 43, 60, 3415, 3460, 3460
\__enumext_parse_keys_vii:n . 43, 60, 4344, 4386,
    4386
\__enumext_parse_keys_viii:n . 4613, 4657, 4657
\__enumext_parse_save_key:n 70, 2238, 2243, 2243
\__enumext_parse_save_key_vii:n 70, 2233, 2243,
    2251
\__enumext_parse_series:n 60, 95, 112, 1726, 1726,
    3468, 4392
\__enumext_parse_store_keys:n ..... 95
\l__enumext_parsep_i_skip ..... 1111, 1113
\l__enumext_parsep_ii_skip ..... 1117, 1119
\l__enumext_parsep_iii_skip ..... 1123, 1125
\l__enumext_parsep_vii_skip ..... 4566
\l__enumext_parsep_viii_skip ..... 4820
\l__enumext_partopsep_v_skip . 1166, 1170, 1362,
    1385

```

```

\l__enumext_partopsep_viii_skip . . . . . 1463
\__enumext_phantomsection: 35, 410, 439, 443, 459
\__enumext_pre_itemsep_skip: 50, 51, 1195, 1224,
    1224
\__enumext_print_footnote: . . . 4039, 4062, 4580,
    4850
\__enumext_print_keyans_box:NN 73, 2362, 2362,
    2375, 2519, 2522, 3046, 3086, 4754, 4769
\l__enumext_print_keyans_i_tl . . . . 4915, 4937
\l__enumext_print_keyans_ii_tl . . . 4919, 4938
\l__enumext_print_keyans_iii_tl . . 4923, 4939
\l__enumext_print_keyans_iv_tl . . . 4927, 4940
\l__enumext_print_keyans_starred_tl 123, 124,
    128, 4911, 4959
\l__enumext_print_keyans_vii_tl 123, 4931, 4941
\l__enumext_print_keyans_X_tl . . . . . 128
\__enumext_printkeyans:nnn 124, 4942, 4945, 4945
\__enumext_redefine_item: . 88, 3146, 3146, 3365
\l__enumext_ref_key_arg_tl 38, 50, 226, 603, 604,
    617, 648, 651, 662, 668, 679, 720, 721, 732
\l__enumext_ref_the_count_tl . 38, 50, 610, 613,
    616, 656, 658, 661, 673, 675, 678, 726, 728, 731
\__enumext_regex_counter_style: . . 30, 38, 221,
    221, 611, 657, 674, 727
\__enumext_register_counter_style:Nn . . 479,
    479, 484, 485, 486, 487, 488
\__enumext_remove_extra_parsep_vii: . . 4364,
    4593, 4593
\__enumext_remove_extra_parsep_viii: . 4628,
    4862, 4862
\__enumext_renew_footnote: . . . 4039, 4043, 4557,
    4795
\l__enumext_renew_the_count_v_tl 729, 738, 740
\l__enumext_renew_the_count_vii_tl 659, 688,
    690
\l__enumext_renew_the_count_viii_tl 676, 695,
    697
\l__enumext_renew_the_count_X_tl . . . . . 50
\__enumext_rescan_anskey_env:n . . 80, 82, 2756,
    2851, 2859
\__enumext_reset_global_bool: . . 342, 345, 354
\__enumext_reset_global_int: . . . 342, 344, 348
\__enumext_reset_global_tl: . . . . 342, 346, 360
\__enumext_reset_global_vars: . 33, 83, 342, 342,
    2889
\l__enumext_resume_active_bool 60, 62, 61, 1730,
    1850
\__enumext_resume_counter: . 62, 1848, 1854, 1861
\__enumext_resume_counter:n . 60, 62, 1819, 1824,
    1848, 1848, 1918, 1926
\__enumext_resume_counter_save_ans: . . 62, 63,
    1848, 1859, 1891
\__enumext_resume_counter_series: 62, 63, 1848,
    1857, 1874
\g__enumext_resume_int . . . 61, 1771, 1865, 1866
\__enumext_resume_last:n . . 60, 1726, 1732, 1745
\l__enumext_resume_name_tl 61, 1767, 1775, 1778,
    1794, 1802, 1805, 1851, 1852, 1880, 1887
\__enumext_resume_save_counter: . . 61, 97, 113,
    1758, 1758, 3596, 4411
\__enumext_resume_series:n . 62, 1694, 1815, 1815
\__enumext_resume_starred: . 63, 1695, 1912, 1912
\g__enumext_resume_vii_int 61, 1798, 1870, 1871
\l__enumext_rightmargin_vii_dim . . 4090, 4094,
    4099
\l__enumext_rightmargin_viii_dim . 4121, 4125,
    4130
\__enumext_safe_exec: . . 34, 94, 3414, 3449, 3449
\__enumext_safe_exec_vii: . 34, 4343, 4369, 4369
\__enumext_safe_exec_viii: 117, 4612, 4633, 4633
\l__enumext_series_name_tl . . . . . 62
\l__enumext_series_str . . 61, 95, 112, 1692, 1728,
    1736, 1737, 1739, 1741, 1762, 1765, 1769, 1789, 1792,
    1796, 3464, 4390
\__enumext_set_error:nn . . . . . 5067, 5077, 5079
\__enumext_set_item_width: . 94, 3424, 3432, 3432
\__enumext_set_parse:n . . . . . 5051, 5067, 5067
\l__enumext_setkey_tmpa_int . . . 119, 5044, 5048
\l__enumext_setkey_tmpa_seq . . . 119, 5042, 5052,
    5058, 5060, 5062, 5074
\l__enumext_setkey_tmpa_tl . . . . 119, 5050, 5054
\l__enumext_setkey_tmpb_seq . . . 119, 5043, 5046,
    5050, 5051
\l__enumext_setkey_tmpb_tl 119, 5069, 5071, 5072
\l__enumext_show_answer_bool . 2178, 2202, 2530,
    3016, 3030, 3993, 4752
\__enumext_show_length:nnn . . 45, 229, 229, 5288,
    5289, 5290, 5291, 5292, 5293, 5294, 5295, 5296, 5297,
    5303, 5304, 5305, 5306, 5307, 5308, 5309, 5310, 5311,
    5312
\l__enumext_show_position_bool . . . 2181, 2205,
    2534, 3020, 3031, 3994, 4756
\g__enumext_standar_bool 31, 94, 34, 261, 264, 283,
    357, 1760, 1825, 1837, 1863, 1876, 1914, 2054, 2068,
    2399, 2412, 2427, 3484
\l__enumext_standar_bool . 94, 97, 34, 2400, 3456,
    3595, 4383
\l__enumext_standar_first_bool 31, 95, 34, 288,
    856, 1747, 1894, 1956, 1963
\__enumext_standar_item_vii:w 114, 4455, 4457,
    4457
\__enumext_standar_item_viii:w 119, 4698, 4700,
    4700
\__enumext_standar_ref: . . . . 39, 601, 621, 3367
\__enumext_standar_ref:n . . . . 38, 593, 601, 601
\g__enumext_standar_series_tl . 61, 1749, 1750,
    1916, 1919
\__enumext_standar_unknown_keys:n 3229, 3233,
    3237
\__enumext_standar_unknown_keys:nn 3229, 3239,
    3241
\g__enumext_starred_bool 31, 112, 34, 271, 274, 298,
    358, 1787, 1830, 1841, 1868, 1883, 1922, 2028, 2074,
    2390, 2930, 4285
\l__enumext_starred_bool 112, 113, 118, 34, 1511,
    2428, 2463, 2469, 2517, 2805, 2810, 3039, 3052, 3457,
    4382, 4410, 4645, 4649
\__enumext_starred_columns_set_vii: . . 4072,
    4072, 4354
\__enumext_starred_columns_set_viii: . 4072,
    4103, 4620
\l__enumext_starred_first_bool 31, 112, 34, 303,
    867, 1752, 1903, 1956, 1963
\__enumext_starred_item:nn . . . 3109, 3109, 3152
\__enumext_starred_item_exec: . 120, 4744, 4744,
    4799
\__enumext_starred_item_vii:w 114, 4454, 4471,
    4471

```

```

\__enumext_starred_item_vii_aux_i:w .. 4471,
4476, 4479
\__enumext_starred_item_vii_aux_ii:w . 4471,
4477, 4482, 4484
\__enumext_starred_item_vii_aux_iii:w 4471,
4487, 4494
\__enumext_starred_item_viii:w 119, 120, 4697,
4714, 4714
\__enumext_starred_item_viii_aux_i:w .. 120,
4714, 4719, 4722
\__enumext_starred_item_viii_aux_ii:w . 120,
4714, 4720, 4737, 4739
\__enumext_starred_joined_item_vii:n 108, 114,
4134, 4134, 4452
\__enumext_starred_joined_item_viii:n . 108,
119, 4134, 4183, 4695
\__enumext_starred_ref: . . . . 40, 646, 684, 3397
\__enumext_starred_ref:n . . . . 39, 640, 646, 646
\g__enumext_starred_series_tl . 61, 1754, 1755,
1924, 1927
\__enumext_starred_unknown_keys:n 3211, 3213,
3215
\__enumext_starred_unknown_keys:nn 3211, 3217,
3219
\__enumext_start_from:NNn 41, 743, 743, 756, 778,
784
\l__enumext_start_i_int . . . . . 1866, 1878, 1897
\__enumext_start_item_tmp_vii: 111, 4357, 4437,
4437
\__enumext_start_item_tmp_viii: .. 117, 4623,
4680, 4680
\__enumext_start_item_vii:w 114, 116, 4463, 4468,
4491, 4498, 4544, 4544
\__enumext_start_item_viii:w .. 119, 4706, 4711,
4742, 4772, 4772
\g__enumext_start_line_tl 31, 34, 291, 306, 363,
2098, 2103, 2108, 2122, 2127, 2132
\__enumext_start_list:nn .. 33, 91, 103, 381, 383,
3418, 3604, 4347, 4615
\__enumext_start_list_tag:n .. 3732, 3756, 4560
\__enumext_start_mini_vii: 112, 4232, 4232, 4402
\__enumext_start_mini_viii: . . . 118, 4287, 4287,
4668
\__enumext_start_save_ans_msg: 64, 1940, 1940,
1965
\__enumext_start_store_level: . 95, 3417, 3478,
3478
\__enumext_start_store_level_vii: 113, 4346,
4413, 4413
\l__enumext_start_vii_int . . . 1871, 1885, 1906
\l__enumext_start_X_int . . . . . 98
\__enumext_stop_item_tmp_vii: .. 111, 113, 116,
4356, 4363, 4439, 4546
\__enumext_stop_item_tmp_viii: 117, 119, 4622,
4627, 4682, 4774
\__enumext_stop_item_vii: 116, 4546, 4569, 4569
\__enumext_stop_item_viii: 122, 4774, 4842, 4842
\__enumext_stop_list: .. 33, 381, 384, 3428, 3615,
4365, 4630
\__enumext_stop_list_tag:n . . . 3732, 3774, 4572
\__enumext_stop_mini_vii: 110, 113, 4232, 4251,
4406
\__enumext_stop_mini_viii: 118, 4287, 4306, 4672
\__enumext_stop_save_ans_msg: . 64, 1940, 1945,
2878
\__enumext_stop_start_list_tag: .. 3732, 3765,
4562
\__enumext_stop_store_level: .. 95, 3429, 3478,
3507
\__enumext_stop_store_level_vii: . 113, 4366,
4413, 4423
\l__enumext_store_active_bool 28, 65, 110, 1895,
1904, 1972, 2603, 3482, 3495, 3637, 3645, 3920, 3953,
4415, 4425, 4635, 4651
\__enumext_store_active_keys:n .. 70, 95, 2211,
2211, 3475
\__enumext_store_active_keys_vii:n . 70, 112,
2211, 2221, 4393
\__enumext_store_addto_prop:n 71, 83, 2286, 2286,
2294, 2454, 2911, 4747
\__enumext_store_addto_seq:n 72, 85, 2295, 2295,
2299, 2306, 2320, 2328, 2337, 2351, 2359, 2512, 3004
\l__enumext_store_anskey_arg_tl .. 28, 75, 110,
2460, 2465, 2467, 2472, 2479, 2482, 2492, 2497, 2500,
2506, 2512
\__enumext_store_anskey_code:n 74, 77, 82, 2451,
2451, 2596, 2849, 2857
\l__enumext_store_anskey_env_tl .. 28, 81, 110,
2779, 2783, 2789, 2851, 2859
\l__enumext_store_anskey_opt_tl 28, 81, 82, 110,
2780, 2807, 2813, 2820, 2826, 2836, 2846, 2855
\__enumext_store_anskey_safe_outer: . . . . 77
\g__enumext_store_columns_break_bool . 2703,
2804, 2866
\l__enumext_store_columns_break_bool . 2462,
2552
\l__enumext_store_current_label_tl 28, 83-85,
120, 110, 2894, 2897, 2900, 2907, 2909, 2911, 2968,
2971, 2974, 2980, 2985, 2995, 3004, 4724, 4729, 4733,
4746, 4747, 4749
\l__enumext_store_current_label_tmp_tl . 28,
110, 3265, 3269
\l__enumext_store_current_opt_arg_tl 28, 120,
110, 3014, 3027, 3033, 4735
\__enumext_store_internal_ref: .. 73, 74, 2376,
2376, 2457
\g__enumext_store_item_join_int .. 2706, 2811,
2815, 2867
\l__enumext_store_item_join_int .. 2470, 2474,
2555
\g__enumext_store_item_star_bool . 2708, 2818,
2868
\l__enumext_store_item_star_bool . 2477, 2557
\g__enumext_store_item_symbol_sep_dim 2713,
2833, 2838, 2870
\l__enumext_store_item_symbol_sep_dim 2489,
2494, 2562
\g__enumext_store_item_symbol_tl . 2711, 2824,
2828, 2869
\l__enumext_store_item_symbol_tl . 2480, 2484,
2560
\l__enumext_store_keyans_item_opt_sep_-
tl . . . . 2164, 2905, 2907, 2978, 2982, 4727, 4731
\__enumext_store_level_close: . 72, 2300, 2324,
3511
\__enumext_store_level_close_vii: . 72, 2331,
2355, 4429
\__enumext_store_level_open: 72, 95, 2300, 2300,
3490, 3503

```


[__enumext_store_level_open_vii](#): .. [72](#), [2331](#), [2331](#), [4419](#)
[\g__enumext_store_name_tl](#) [28](#), [65](#), [110](#), [362](#), [369](#), [370](#), [371](#), [372](#), [1948](#), [1974](#), [2097](#), [2102](#), [2107](#), [2121](#), [2126](#), [2131](#), [2876](#)
[\l__enumext_store_name_tl](#) [28](#), [64](#), [66](#), [110](#), [1781](#), [1784](#), [1808](#), [1811](#), [1899](#), [1908](#), [1943](#), [1952](#), [1953](#), [1974](#), [1975](#), [1976](#), [1978](#), [1979](#), [1981](#), [1983](#), [1984](#), [1986](#), [1988](#), [1989](#), [2013](#), [2288](#), [2290](#), [2297](#), [2440](#), [2441](#), [2542](#), [2785](#), [2951](#), [2952](#), [3065](#), [3078](#), [4764](#)
[\l__enumext_store_ref_key_bool](#) [74](#), [2187](#), [2455](#), [2503](#), [2915](#), [2992](#)
[\l__enumext_store_save_key_vii_bool](#) .. [2223](#), [2253](#)
[\l__enumext_store_save_key_vii_tl](#) [2225](#), [2226](#), [2254](#), [2255](#), [2335](#), [2343](#), [2347](#), [2351](#)
[\l__enumext_store_save_key_X_bool](#) .. [70](#), [128](#)
[\l__enumext_store_save_key_X_tl](#) .. [70](#), [128](#)
[\l__enumext_store_upper_level_X_bool](#) .. [128](#)
[__enumext_storing_exec](#): [64](#), [65](#), [79](#), [1950](#), [1966](#), [1970](#)
[__enumext_storing_set:n](#) .. [64](#), [1935](#), [1950](#), [1950](#)
[\l__enumext_the_counter_v_tl](#) .. [728](#)
[\l__enumext_the_counter_vii_tl](#) .. [658](#)
[\l__enumext_the_counter_viii_tl](#) .. [675](#)
[\l__enumext_the_counter_X_tl](#) .. [50](#)
[__enumext_tmp:n](#) [45](#), [49](#), [54](#), [60](#), [71](#), [78](#), [79](#), [85](#), [92](#), [97](#), [98](#), [109](#), [131](#), [138](#), [163](#), [167](#), [171](#), [191](#), [843](#), [852](#), [1688](#), [1699](#), [1931](#), [1939](#), [1992](#), [2010](#), [2151](#), [2192](#), [2193](#), [2210](#), [2229](#), [2242](#), [2378](#), [2385](#), [2386](#), [2407](#), [2420](#), [2423](#), [2434](#), [2917](#), [2924](#), [3189](#), [3196](#), [3229](#), [3236](#), [3337](#), [3376](#), [3377](#), [3411](#)
[__enumext_tmp:nn](#) [511](#), [532](#), [533](#), [564](#), [565](#), [580](#), [773](#), [798](#), [879](#), [901](#), [902](#), [922](#), [975](#), [983](#), [984](#), [998](#), [1063](#), [1079](#), [1080](#), [1093](#), [1577](#), [1593](#), [3173](#), [3188](#)
[__enumext_tmp:nnn](#) [581](#), [597](#), [598](#), [599](#), [600](#), [628](#), [644](#), [645](#)
[__enumext_tmp:nnnnn](#) [799](#), [824](#), [827](#), [830](#), [832](#), [834](#), [837](#), [840](#)
[__enumext_tmp:w](#) .. [4890](#), [4893](#)
[\l__enumext_tmpa_vii_int](#) [4082](#), [4085](#), [4094](#), [4125](#)
[\l__enumext_tmpa_viii_int](#) .. [4113](#), [4116](#)
[\l__enumext_tmpa_X_dim](#) .. [171](#)
[\l__enumext_tmpa_X_int](#) .. [171](#)
[\l__enumext_topsep_v_skip](#) [1152](#), [1156](#), [1356](#), [3877](#)
[\l__enumext_topsep_vii_skip](#) .. [1433](#), [1442](#), [1446](#)
[\l__enumext_topsep_viii_skip](#) . [1455](#), [1477](#), [1481](#)
[__enumext_undefine_anskey_env](#): [78](#), [83](#), [2636](#), [2636](#), [2887](#)
[__enumext_unskip_unkern](#): .. [31](#), [235](#), [235](#), [1144](#), [1173](#), [1206](#), [1378](#), [3571](#), [3572](#), [3586](#), [3706](#), [3707](#), [3721](#)
[\l__enumext_vspace_a_star_v_bool](#) .. [1626](#)
[\l__enumext_vspace_a_star_vii_bool](#) .. [1648](#)
[\l__enumext_vspace_a_star_viii_bool](#) .. [1659](#)
[\l__enumext_vspace_a_star_X_bool](#) .. [98](#)
[__enumext_vspace_above](#): [57](#), [96](#), [1594](#), [1594](#), [3516](#)
[__enumext_vspace_above_v](#): [58](#), [1622](#), [1622](#), [3661](#)
[\l__enumext_vspace_above_v_skip](#) .. [1624](#), [1628](#), [1630](#)
[__enumext_vspace_above_vii](#): [58](#), [112](#), [1644](#), [1644](#), [4399](#)
[\l__enumext_vspace_above_vii_skip](#) [1646](#), [1650](#), [1652](#)
[__enumext_vspace_above_viii](#): [58](#), [1644](#), [1655](#), [4666](#)
[\l__enumext_vspace_above_viii_skip](#) [1657](#), [1661](#), [1663](#)
[\l__enumext_vspace_b_star_v_bool](#) .. [1637](#)
[\l__enumext_vspace_b_star_vii_bool](#) .. [1670](#)
[\l__enumext_vspace_b_star_viii_bool](#) .. [1681](#)
[\l__enumext_vspace_b_star_X_bool](#) .. [98](#)
[__enumext_vspace_below](#): [57](#), [97](#), [1608](#), [1608](#), [3594](#)
[__enumext_vspace_below_v](#): [58](#), [1633](#), [1633](#), [3730](#)
[\l__enumext_vspace_below_v_skip](#) .. [1635](#), [1639](#), [1641](#)
[__enumext_vspace_below_vii](#): [59](#), [113](#), [1666](#), [1666](#), [4409](#)
[\l__enumext_vspace_below_vii_skip](#) [1668](#), [1672](#), [1674](#)
[__enumext_vspace_below_viii](#): [59](#), [1666](#), [1677](#), [4674](#)
[\l__enumext_vspace_below_viii_skip](#) [1679](#), [1683](#), [1685](#)
[__enumext_widest_from:nnnn](#) .. [41](#), [757](#), [757](#), [772](#), [791](#)
[\g__enumext_widest_label_tl](#) [26](#), [36](#), [67](#), [499](#), [503](#), [507](#)
[\l__enumext_wrap_label_opt_v_bool](#) .. [3259](#)
[\l__enumext_wrap_label_opt_vii_bool](#) [114](#), [4462](#)
[\l__enumext_wrap_label_opt_viii_bool](#) .. [119](#), [4705](#)
[\l__enumext_wrap_label_opt_X_bool](#) .. [98](#)
[\l__enumext_wrap_label_v_bool](#) [3255](#), [3259](#), [3267](#), [3297](#)
[\l__enumext_wrap_label_vii_bool](#) .. [114](#), [4462](#), [4466](#), [4474](#), [4528](#)
[\l__enumext_wrap_label_viii_bool](#) .. [119](#), [4705](#), [4709](#), [4717](#), [4801](#)
[\l__enumext_wrap_label_X_bool](#) .. [98](#)
[__enumext_wrapper_label_v:n](#) .. [3299](#), [4004](#)
[__enumext_wrapper_label_vii:n](#) .. [4533](#)
[__enumext_wrapper_label_viii:n](#) .. [4806](#)
[\l__enumext_write_aux_file_tl](#) .. [29](#), [74](#), [84](#), [160](#), [2443](#), [2449](#), [2958](#), [2964](#)
enumext* .. [5](#), [4341](#)
enumXi .. [471](#)
enumXii .. [471](#)
enumXiii .. [471](#)
enumXiv .. [471](#)
enumXv .. [471](#)
enumXvi .. [471](#)
enumXvii .. [471](#)
enumXviii .. [471](#)
Environments provide by [enumext](#):
 anskey* [28](#), [65](#), [73](#), [74](#), [76](#), [78](#), [79](#), [81](#), [83](#), [95](#), [113](#), [124](#), [129](#), [131](#)
 enumext* [25](#), [26](#), [29–31](#), [34](#), [36](#), [39](#), [40](#), [42–45](#), [47](#), [54](#), [55](#), [58–64](#), [66](#), [67](#), [69–78](#), [81](#), [83](#), [84](#), [88](#), [89](#), [93–95](#), [100](#), [106–108](#), [110](#), [113](#), [116–119](#), [121](#), [123–125](#), [127](#), [130](#), [133](#), [135](#)
 enumext [25](#), [26](#), [30](#), [31](#), [34](#), [36–40](#), [42–50](#), [53](#), [55–57](#), [59–64](#), [66](#), [67](#), [69–78](#), [81](#), [83](#), [84](#), [87](#), [88](#), [90–92](#), [94](#), [95](#), [98](#), [99](#), [102](#), [107](#), [109](#), [112](#), [113](#), [118](#), [123–125](#), [127](#), [130](#), [131](#), [133](#)
 keyans* [25](#), [26](#), [28–32](#), [36](#), [39–42](#), [44](#), [45](#), [47](#), [54](#), [55](#), [58](#), [59](#), [65](#), [68](#), [69](#), [71](#), [79](#), [83](#), [89](#), [93](#), [100](#), [106–108](#), [118](#), [130](#), [132](#), [135](#)
 keyanspic [25](#), [26](#), [28](#), [29](#), [32](#), [36](#), [37](#), [40](#), [65](#), [68](#), [71](#), [72](#), [79](#), [83–85](#), [100–103](#), [105](#), [132](#)
 keyans [25](#), [26](#), [28](#), [29](#), [31](#), [32](#), [36](#), [37](#), [40](#), [42](#), [44](#), [45](#), [47](#), [49](#),

53, 55–58, 65, 68, 69, 71, 72, 79, 83–85, 89–92, 98, 99,
102, 103, 105, 109, 119, 130, 132

Environments:

list 30, 33, 91, 94, 100, 102
lrbox 116
minipage ... 30, 34, 47, 50, 51, 102–104, 107, 116, 122
multicols 48–51, 56, 96, 97
scontents 79, 81

exp commands:

\exp_after:wN 4893
\exp_args:Ne 2848, 2856, 3472, 4881
\exp_args:NV ... 2568, 2723, 3199, 3217, 3239, 5165
\exp_not:N . 58, 502, 616, 661, 678, 731, 932, 946, 947,
958, 959, 970, 971, 2508, 2539, 2540, 2997, 3062, 3063,
3075, 3076, 4761, 4762, 4890
\exp_not:n 293, 308, 321, 329, 337, 555, 575, 616, 617,
661, 662, 678, 679, 731, 732, 933, 1715, 1724, 2175,
2272, 2284, 2446, 2474, 2484, 2494, 2508, 2509, 2815,
2828, 2838, 2961, 2999, 3001, 4994, 5004, 5197, 5202

F

\fbox 2158
\fboxrule 2158
\fboxsep 2158

file commands:

\file_input_stop: 5601
first 984
font 511
\footnote 106
\footnote 106, 4047
\footnotemark 4057
\footnotesize 2540, 3063, 3076, 4762
\footnotetext 4041
\foreachkeyans 16, 127, 5130

G

\getkeyans 16, 123, 4879
group commands:
\group_begin: .. 2538, 2583, 2758, 2845, 3061, 3074,
4760, 4936
\group_end: 2545, 2599, 2862, 3068, 3081, 4767, 4943

H

\hbadness 4574, 4846
hbox commands:
\hbox_overlap_left:n 3142, 4524
\hbox_set:Nn 491
\hbox_set_end: 4573, 4845
\hbox_set_to_wd:Nnw 4547, 4787
\hfill 541, 545, 550, 551, 1533, 1560, 2508, 2997, 4256, 4311
hook commands:
\hook_gput_code:nnn 9, 201, 205, 209, 408
\hook_gremove_code:nn 81, 2774
\hook_gset_rule:nnnn 409
\hook_if_empty:nTF 2772
\hyperlink 75, 85
\hyperlink 2508, 2997
\hypertarget 35
\hypertarget 438

I

\IfDocumentMetadataTF 3758, 3767, 3776, 3813, 3822, 3831,
3885, 3895, 3904, 3914, 3916, 3936, 3941, 4362, 4553,
4576
\IfHyperBoolean 416
\IfPackageLoadedTF 11, 19, 412, 426

\ignorespaces 935, 4358, 4624
\inputlineno 293, 308, 321, 329, 337
int commands:

\int_add:Nn 4167, 4216
\int_case:nn ... 1108, 1226, 2023, 2049, 2088, 2112
\int_case:nnTF 237
\int_compare:nNnTF .. 394, 649, 666, 686, 693, 1196,
1215, 1369, 1387, 1499, 1515, 1527, 1555, 2136, 2142,
2607, 2611, 2615, 2623, 2669, 2673, 2677, 2874, 2895,
2934, 2939, 2944, 2969, 3057, 3454, 3465, 3487, 3500,
3537, 3552, 3567, 3580, 3646, 3650, 3678, 3703, 3715,
3841, 3957, 3961, 4137, 4147, 4163, 4186, 4196, 4212,
4374, 4378, 4417, 4427, 4583, 4595, 4640, 4652, 4852,
4864, 5048, 5180
\int_compare_p:nNn ... 262, 272, 284, 285, 299, 300,
1505, 1506, 2029, 2055, 2391, 2401, 2413, 2414, 2429,
2470, 2646, 2647, 2658, 2659, 2811, 3497
\int_decr:N 4166, 4215
\int_eval:n .. 379, 786, 2290, 2441, 2540, 2952, 3063,
3076, 3352, 3396, 4155, 4204, 4762
\int_from_alph:n 751, 765
\int_from_roman:n 753, 767
\int_gadd:Nn 4168, 4217
\int_gdecr:N 2032, 2037, 2041, 2045, 2058
\int_gincr:N 1865, 1870, 2453, 3007, 3096, 3130, 3273,
3529, 3670, 3991, 4441, 4510, 4684, 4751
\int_gset:Nn 2081, 4055
\int_gset_eq:NN 1764, 1771, 1777, 1783, 1791, 1798,
1804, 1810, 4052
\int_gzero:N . 350, 351, 352, 1541, 1568, 2148, 2867,
3585, 3720, 4606, 4876
\int_if_exist:Ntf 1739, 1775, 1781, 1802, 1808, 1986
\int_incr:N 2622, 3453, 3641, 3840, 4373, 4440, 4639,
4683
\int_mod:nn 4597, 4866
\int_new:N . 28, 29, 30, 31, 32, 33, 61, 62, 86, 102, 121,
141, 142, 153, 154, 155, 157, 168, 174, 175, 176, 177,
178, 1741, 1989
\int_set:Nn 747, 751, 753, 1878, 1885, 1897, 1906, 2759,
3930, 3931, 4082, 4113, 4136, 4142, 4158, 4185, 4191,
4207, 4574, 4846, 5044, 5182
\int_set_eq:NN 1866, 1871, 4165, 4214
\int_sign:n 2083
\int_step_function:nnN 2407, 2420, 2434
\int_step_function:nnnN 5186
\int_step_inline:nn 5096
\int_step_inline:nnn 3932
\int_to_roman:n 213, 2387, 2424
\int_use:N 372, 377, 378, 1197, 1216, 1528, 1880, 1887,
1899, 1908, 3352, 3371, 3396, 3473, 3538, 3547, 3562,
3568, 4140, 4141, 4153, 4189, 4190, 4202, 5517, 5521,
5527, 5531
\int_zero:N 4587, 4856
\item . 87, 90, 113, 116, 119, 121, 385, 2308, 2314, 2339, 2345,
2467, 2971, 2974, 3148, 3277, 3889, 3891, 4355, 4357,
4361, 4621, 4623, 4749, 4831
\item* 5, 14, 68, 3275
item-pos* 3173
item-sym* 3173
\itemindent 92
\itemindent 91
itemindent 879
\itemsep 103
\itemsep 3866, 3872

`\itemwidth` . 461, 2158, 3434, 3443, 3620, 3629, 4176, 4180, 4225, 4229

K

`keyans` 14, 3599
`keyans*` 14, 4610
`keyanspic` 15, 3881

Keys for `\anskey` provide by `enumext`:

`break-col` 75, 76, 79–81
`item-join` 75, 76, 79–81
`item-pos*` 75, 76, 79, 80, 82
`item-star` 75, 76, 79, 80, 82
`item-sym*` 75, 76, 79, 80, 82

Keys for `anskey*` provide by `enumext`:

`break-col` 75, 76, 79–81
`item-join` 75, 76, 79–81
`item-pos*` 75, 76, 79, 80, 82
`item-star` 75, 76, 79, 80, 82
`item-sym*` 75, 76, 79, 80, 82

Keys for environments provide by `enumext`:

`above*` 27, 57, 58, 96, 112
`above` 27, 57, 58, 96, 112, 118
`after` 45, 46, 97, 113, 118
`align` 27, 37, 88, 91, 115, 129
`base-fix` 43, 59, 71, 95, 112, 124
`before*` 45, 46, 96, 112, 118
`before` 45, 46
`below*` 27, 57–59, 97, 113
`below` 27, 57–59, 97, 113, 118
`check-ans` . 29, 30, 32, 64–68, 71, 82, 85, 97, 98, 113, 117, 131
`columns-sep` 47, 96
`columns` 27, 47, 57, 96
`first` 45, 46, 116
`font` 37, 88, 91, 115
`item-pos*` 87, 88
`item-sym*` 28, 87, 88
`itemindent` 27, 44, 87, 91, 116
`itemsep` 42, 93
`labelsep` 37, 92, 115
`labelwidth` 36–41, 92, 115
`label` 26, 36, 38, 41, 107
`lisparindent` 93
`list-indent` 27, 44, 103
`list-offset` 44, 94, 98
`listparindent` 44, 116
`mark-ans` 69, 71, 76
`mark-pos` 69, 129
`mark-ref` 69, 71, 73, 75
`mini-env` 27, 34, 47, 56, 57, 71, 96, 109, 110, 112, 113, 118
`mini-right*` 27, 30, 47, 71, 110, 112, 113
`mini-right` 27, 30, 47, 55, 71, 110, 112, 113
`mini-sep` 27, 47, 71, 96
`no-store` 29, 64–66, 71, 77, 87
`noitemsep` 42
`nosep` 42
`parindent` 93
`parsep` 42, 93, 116
`partopsep` 42
`ref` 26, 30, 38–40, 130
`resume*` 26, 59, 60, 63–65, 71, 97, 113, 125
`resume` 26, 33, 59–65, 71, 97, 113, 125
`rightmargin` 44, 107
`save-ans` 28, 33, 60–64, 66, 67, 70–72, 77–79, 82–84, 90, 98, 105, 117, 119, 120, 123, 125, 131

`save-key` 28, 60, 70, 95, 112
`save-pos` 71
`save-ref` 29, 35, 69, 71, 73–75, 83, 85, 90, 120
`save-sep` 69, 71, 120
`series` 26, 59–63, 71, 95, 97, 112, 113, 125
`show-ans` 69, 71, 73, 74, 76, 90, 120
`show-length` 31, 45, 130
`show-pos` 28, 69, 73, 74, 76, 85, 90, 120
`start*` 27, 41, 42, 60
`start` 27, 30, 41, 42, 60
`store-key` 70
`topsep` 42
`widest` 26, 30, 41, 42
`wrap-ans` 35, 69, 71, 73, 76
`wrap-label*` 27, 37, 87, 88, 91, 114, 115, 119
`wrap-label` 27, 37, 87, 88, 91, 114, 115, 119
`wrap-opt` 69, 71

keys commands:

`\keys_define:nn` 513, 535, 567, 583, 630, 701, 775, 801, 845, 881, 904, 977, 986, 1065, 1082, 1579, 1690, 1933, 1994, 2153, 2195, 2231, 2236, 2550, 2701, 2737, 3175, 3191, 3211, 3231, 4907, 5006, 5122, 5130
`\keys_if_exist_p:nn` 5118, 5119
`\l_keys_key_str` 77, 80, 2568, 2723, 3199, 3217, 3239, 5165, 5273
`\keys_precompile:nnN` . 124, 197, 197, 4909, 4913, 4917, 4921, 4925, 4929, 5148
`\keys_set:nn` . 527, 861, 872, 1088, 1584, 1589, 1827, 1832, 1919, 1927, 2588, 3467, 3472, 3657, 4391, 4661, 4961, 4968, 5010, 5015, 5016, 5017, 5018, 5021, 5026, 5027, 5028, 5029, 5030, 5031, 5032, 5064, 5174
`\keys_set_known:nn` 2855

keyval commands:

`\keyval_parse:NNn` 1704, 2261, 4982

L

`label` 581, 628, 701

Labels provide by `enumext`:

`\Alph*` 36
`\Roman*` 36
`\alph*` 36
`\arabic*` 30, 36
`\roman*` 36
`\labelsep` 103
`\labelsep` 3867, 3870
`labelsep` 511
`\labelwidth` 36, 103
`\labelwidth` 3867, 3868
`labelwidth` 511
`\lastkern` 248
`\lastnodetype` 237
`\lastskip` 242
`\leftmargin` 92
`\leftmargin` 91, 3867

legacy commands:

`\legacy_if:nTF` 4502, 4505, 4775, 4778
`\legacy_if_gset_false:n` 399
`\legacy_if_set_false:n` 4504, 4777
`\legacy_if_set_true:n` 4467, 4490, 4497, 4710, 4741, 4782

`\linewidth` 96
`\linewidth` 3436, 3524, 3622, 3667, 3929, 4085, 4116, 4238, 4293
`\list` 383
`list-indent` 879

list-offset	879
\listparindent	3869
listparindent	879
M	
\makebox	107
\makebox	2366, 2368, 4001, 4530, 4537, 4803, 4810
\makelabel	87, 88, 91, 106
\makelabel	87, 90, 3159, 3293
\makesavenoteenv	432
mark-ans	2151
mark-pos	2151, 2193
mark-ref	2151
mini-env	1063
mini-sep	1063
\minipage	389
\miniright	10, 55, 1497, 1545, 1572, 3583, 3718
mode commands:	
\mode_if_math:TF	2631, 2685
\mode_if_vertical:TF	1133, 1162, 1183, 1207, 1358, 1379
\mode_leave_vertical:	859, 870, 932, 946, 958, 970, 2364, 3140, 4522
msg commands:	
\msg_error:nn	1547, 1574, 2592, 2625, 2629, 2683, 2791, 3648, 3652, 3843, 3893, 3959, 4376, 4642, 4654, 5033, 5092
\msg_error:nnn	606, 653, 670, 723, 1501, 1508, 1513, 1543, 1570, 1839, 1843, 1958, 2574, 2633, 2651, 2663, 2671, 2675, 2679, 2687, 2729, 3205, 3223, 3245, 4380, 4647, 4895, 4904, 4975, 5080, 5111, 5120, 5157, 5178
\msg_error:nnnn	2577, 2605, 2609, 2613, 2617, 2732, 3208, 3226, 3248, 3639, 3955, 3963, 4637, 4956, 5160
\msg_error:nnnnn	554, 574, 2174
\msg_fatal:nn	3455
\msg_fatal:nnn	465
\msg_info:nnn	13, 16, 21, 24, 414, 428
\msg_line_context:	5238, 5243, 5248, 5277, 5282, 5287, 5302, 5317, 5321, 5325, 5329, 5333, 5337, 5344, 5351, 5357, 5371, 5375, 5380, 5384, 5388, 5392, 5397, 5401, 5405, 5409, 5414, 5449, 5453, 5458, 5463, 5467, 5472, 5548, 5552, 5557, 5562, 5567, 5571, 5575, 5579, 5583, 5587, 5591, 5595, 5599
\msg_log:nnn	1978, 1983, 1988
\msg_log:nnnnn	376, 2121, 2126, 2131
\msg_log:nnnnnn	368
\msg_new:nnn	5205, 5209, 5213, 5217, 5222, 5235, 5240, 5245, 5250, 5259, 5267, 5271, 5275, 5280, 5285, 5300, 5315, 5319, 5323, 5327, 5331, 5335, 5339, 5348, 5354, 5360, 5364, 5368, 5373, 5378, 5382, 5386, 5390, 5395, 5399, 5403, 5407, 5412, 5447, 5451, 5456, 5461, 5465, 5470, 5546, 5550, 5555, 5560, 5565, 5569, 5573, 5577, 5581, 5585, 5589, 5593, 5597
\msg_new:nnnn	5226, 5417, 5426, 5435, 5441, 5474, 5484, 5494, 5504, 5514, 5524, 5534, 5540
\msg_term:nnnn	1942, 1947, 3361, 3371, 3402, 3407
\msg_term:nnnnn	2102
\msg_warning:nn	3582, 3717
\msg_warning:nnnn	2139, 2145, 3309, 3314, 4139, 4152, 4188, 4201
\msg_warning:nnnnn	2097, 2107
\multicolsep	96
\multicolsep	1200, 1372, 3558, 3694
N	
\NeedsTeXFormat	3

\NewCommandCopy	385
\newcounter	468
\NewDocumentCommand	1497, 2580, 3951, 4879, 4934, 5040, 5089, 5167
\NewDocumentEnvironment	3412, 3599, 3881, 4341, 4610
\newenvsc	2694
\newlabel	35
\newlabel	450
no-store	1992
\noindent	4247, 4302, 4586, 4855
\nointerlineskip	1209, 1212, 1381, 1384, 1535, 1562, 4247, 4302
noitemsep	799
\nopagebreak	1145, 1174, 1209, 1212, 1381, 1384, 1488, 1494
\normalfont	2539, 3062, 3075, 4761
nosep	799
P	
Packages:	
caption	110
enumext	25, 35, 38, 64, 92, 102, 129
enumitem	36
expl3	106
footnotehyper	35
hyperref	29, 30, 34, 35, 75, 85, 129
lua-visual-debug	50
multicol	25, 129
scontents	25, 78, 79
shortlst	106, 111, 116
\par	1145, 1174, 1212, 1384, 1488, 1494, 1530, 1535, 1557, 1562, 2516, 3573, 3708, 3723, 3919, 3945, 3948, 4265, 4279, 4320, 4334, 4586, 4855
para commands:	
\para_end:	4603, 4873
\parbox	2158
\parindent	4565, 4819
\parsep	48, 103
\parsep	3393, 3866, 3873, 3878, 3919
parsep	799
\parskip	4566, 4820
\partopsep	103
\partopsep	3394, 3871
partopsep	799
peek commands:	
\peek_meaning:NTF	4446, 4460, 4475, 4486, 4689, 4703, 4718
\peek_meaning_remove:NTF	4453, 4696
\peek_remove_spaces:n	3281
\phantomsection	35
\phantomsection	439
prg commands:	
\prg_do_nothing:	443
\prg_new_protected_conditional:Npnn	215
\prg_replicate:nn	232
\prg_return_false:	219
\prg_return_true:	218
\printkeyans	16, 123, 4934
prop commands:	
\prop_const_from_keyval:Nn	5081
\prop_count:N	370, 2290, 2441, 2542, 2952, 3065, 3078, 4764, 5183
\prop_get:NnNTF	5107
\prop_gput_if_not_in:Nnn	2288
\prop_if_exist:NTF	1976, 4899, 5176
\prop_item:Nn	4901, 5200

\prop_new:N	1979
\ProvidesExplPackage	4
R	
\raggedcolumns	3561, 3697
\raisebox	4016
\ref	73, 83
ref	581, 628, 701
\refstepcounter	4507, 4784
regex commands:	
\regex_match:nnTF	217, 750, 752, 764, 766, 2787
\regex_replace_once:nnN	225
\renewcommand	616, 661, 678, 731
\RenewDocumentCommand	1545, 1572, 3148, 3159, 3277, 3293, 3891, 4047, 4831
\RequirePackage	17, 25
resume	1688
resume*	1688
rightmargin	879
\Roman	36, 41
\Roman	487
\roman	36, 41
\roman	488, 599, 4924
S	
\s	2788
save-ans	1931
save-key	2229
save-ref	2151
save-sep	2151
scan commands:	
\scan_stop:	103, 3889, 4355, 4621, 4890, 4893
scontents internal commands:	
\l_scontents_fname_out_tl	2747
__scontents_parse_environment_keys:n	2753
__scontents_rescan_tokens:n	2760
\l_scontents_storing_bool	2745
\l_scontents_writing_bool	2746
seq commands:	
\seq_clear:N	5042, 5185
\seq_const_from_clist:Nn	5035
\seq_count:N	371, 3911, 5046
\seq_gclear:N	4045, 4046
\seq_gput_right:Nn	2297, 4058, 4059
\seq_if_empty:N	4064, 4949, 5060
\seq_if_exist:N	1981, 4947
\seq_if_in:Nn	4954
\seq_item:Nn	2785, 3939
\seq_map_function:NN	5051
\seq_map_inline:Nn	4962, 4969, 5061, 5062
\seq_map_pairwise_function:NNN	4066
\seq_new:N	122, 123, 125, 139, 169, 170, 1984
\seq_pop_left:NN	5050
\seq_put_right:Nn	3965, 5058, 5074, 5195
\seq_set_from_clist:Nn	5043
\seq_set_map_e:NNn	5052
\seq_show:N	4951
\seq_use:Nn	197, 198, 5191
series	1688
\setcounter	761, 765, 767, 3352, 3396, 3918
\setenumext	6, 125, 5040
\setenumextmeta	6, 127, 5081
show-ans	2151, 2193
show-length	975
show-pos	2193

skip commands:

```

\skip_add:Nn 1113, 1119, 1125, 1135, 1139, 1164, 1168,
1185, 1243, 1245, 1259, 1262, 1283, 1285, 1299, 1302,
1322, 1324, 1338, 1341, 1360, 1409, 1410, 1421, 1423,
3866
\skip_gset:Nn . . . . . 1436, 1440, 1444
\skip_gzero_new:N . . . . . 1431, 1432
\skip_horizontal:N 947, 959, 971, 4525, 4542, 4590,
4816, 4859
\skip_horizontal:n . . . 933, 2365, 2373, 3141, 3143,
4435, 4523, 4678, 4825
\skip_if_eq:nnTF 1111, 1117, 1123, 1229, 1269, 1309,
1397, 1433, 1455, 1596, 1610, 1624, 1635, 1646, 1657,
1668, 1679
\skip_new:N . . . 81, 82, 83, 87, 88, 89, 90, 91, 143, 189
\skip_set:Nn 1096, 1100, 1150, 1154, 1179, 1232, 1233,
1251, 1272, 1273, 1291, 1311, 1312, 1330, 1354, 1400,
1401, 1415, 1435, 1439, 1457, 1461, 1465, 1471, 1475,
1479, 3859, 3874
\skip_set_eq:NN 1190, 1191, 1193, 1200, 1365, 1366,
1367, 1372, 3350, 3392, 3393, 4566, 4820
\skip_sub:Nn 1239, 1241, 1255, 1257, 1279, 1281, 1295,
1297, 1318, 1320, 1334, 1336, 1407, 1408, 1419, 1420
\skip_use:N 1098, 1102, 1137, 1141, 1146, 1166, 1170,
1181, 1187, 1597, 1601, 1604, 1611, 1615, 1618, 3573
\skip_vertical:N . . . . . 400, 403, 4605, 4875
\skip_vertical:n . . . . . 4604, 4874
\skip_zero:N 1199, 1213, 1351, 1352, 1353, 1371, 1385,
3394, 3558, 3694, 3871, 3872
\skip_zero_new:N . . . . . 1430, 1452, 1453, 1454
\l_tmpa_skip 1251, 1261, 1264, 1291, 1301, 1304, 1330,
1340, 1343, 1415, 1422, 1424
\c_zero_skip . 400, 403, 1111, 1117, 1123, 1270, 1309,
1433, 1455, 1597, 1611, 1624, 1635, 1646, 1657, 1668,
1679, 4605, 4875
\small . . . . . 4912, 4916, 4920, 4924, 4928, 4932
socket commands:
\socket_assign_plug:nn . . 3760, 3769, 3778, 3815,
3824, 3833
\socket_new:nn . . . . . 3732, 3783
\socket_new_plug:nnn 3733, 3740, 3748, 3784, 3792,
3801
\socket_use:n . . . . . 3761, 3816
\socket_use:nn . . . . . 3770, 3779, 3825, 3834
\star . . . . . 3179
start . . . . . 773
start* . . . . . 773
start-list-tags . . . . . 3732, 3783
\stepcounter . . . . . 4009, 4051
stop-list-tags . . . . . 3732, 3783
stop-start-tags . . . . . 3732, 3783
str commands:
\c_backslash_str 2633, 5238, 5243, 5248, 5253, 5255,
5257, 5262, 5264, 5362, 5366, 5370, 5380, 5384, 5392,
5393, 5397, 5409, 5410, 5414, 5415, 5436, 5438, 5442,
5444, 5472, 5535, 5537, 5541, 5543, 5552, 5553, 5557,
5562, 5563, 5567, 5571, 5575
\c_colon_str . . . . . 2440, 2951, 4890
\c_left_brace_str . . . . . 5343, 5350, 5356
\c_right_brace_str . . . . . 5343, 5350, 5356
\str_case:nn . . . . . 255, 314
\str_case:nnTF . 1711, 1719, 2268, 2276, 4989, 4998
\str_clear:N . . . . . 3464, 4390
\str_count:n . . . . . 232
\str_if_empty:NTF . . . . . 1728, 1769, 1796

```

`\str_if_eq:nnTF` 3353, 3398, 5091
`\str_if_in:nnTF` 4886
`\str_new:N` 129, 145, 184
`\str_set:Nn` 570, 571, 572, 2171, 2172, 2198, 2199, 3849, 3852
`\string` 432
`\strutbox` . 1218, 1221, 1232, 1233, 1244, 1246, 1261, 1264, 1272, 1273, 1284, 1286, 1301, 1304, 1311, 1312, 1323, 1325, 1340, 1343, 1389, 1392, 1400, 1401, 1409, 1410, 1422, 1424, 1435, 1436, 1439, 1446, 1459, 1467, 1473, 1481, 3876, 3919, 3948, 4022
`\SuspendTagging` 4353

T

tag commands:

`\tag_mc_begin:n` 3738, 3790, 3799
`\tag_mc_end:` 3742, 3794, 3803
`\tag_resume:n` 3735, 3786, 3897
`\tag_start:n` 3787, 3906, 3941
`\tag_stop:n` 3808, 3914, 3936
`\tag_struct_begin:n` . 3736, 3737, 3744, 3745, 3746, 3788, 3789, 3796, 3797, 3798, 3907
`\tag_struct_end:` 3916, 4362
`\tag_struct_end:n` 3743, 3750, 3751, 3752, 3753, 3795, 3804, 3805, 3806, 3807
`\tag_suspend:n` 3754, 3809, 3887, 3899
`\tag_tool:n` 3898
`\TeX` and `\LaTeX` commands:

`\auxout` 448
`\@currenvr` 255, 314
`\protected@write` 448

tex commands:

`\tex_newlinechar:D` 2759

text commands:

`\text_expand:n` 4882
`\textasteriskcentered` 2168, 2185
`\the` 242, 248
`\thepage` 454
tl commands:

`\c_space_tl` 3033, 5287, 5302, 5325, 5329, 5516, 5517, 5526, 5527, 5587, 5591
`\tl_clear:N` . . 540, 546, 2149, 2215, 2225, 2246, 2254, 2460, 2779, 2780, 2894, 2968, 4724
`\tl_clear_new:N` 497
`\tl_const:Nn` 50, 481
`\tl_gclear:N` . 362, 363, 364, 1749, 1754, 2869, 3170, 4283, 4338, 4526
`\tl_gclear_new:N` 1736
`\tl_gput_right:Nn` 482
`\tl_greplace_all:Nnn` 503
`\tl_gset:Nn` 290, 291, 305, 306, 1737, 1750, 1755, 1974, 2783, 3117, 4481
`\tl_gset_eq:NN` 499, 3113, 4519
`\tl_if_blank:nTF` 2572, 2590, 2727, 3203, 3221, 3243, 4517, 5155
`\tl_if_empty:NTF` . 604, 623, 651, 668, 688, 695, 721, 738, 1762, 1767, 1789, 1794, 1852, 1916, 1924, 1953, 2013, 2304, 2335, 2480, 2824, 2846, 2876, 2905, 2978, 3027, 3138, 4727, 5072
`\tl_if_empty:nTF` 1817
`\tl_if_exist:NTF` 1822
`\tl_if_novalue:nTF` . . 2586, 2902, 2976, 3012, 3092, 3111, 3119, 3253, 3462, 3909, 4049, 4388, 4659, 4725, 4833
`\tl_map_inline:Nn` 223, 500

`\tl_new:N` 42, 43, 44, 47, 52, 53, 56, 57, 63, 65, 66, 68, 69, 103, 104, 105, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 124, 126, 127, 128, 130, 133, 134, 152, 160, 161, 162, 165, 183
`\tl_put_left::Ne` 2813
`\tl_put_left:Nn` 2312, 2343, 2465, 2807, 2820, 2826, 2836, 3044, 3084, 4268, 4323, 4746, 4749
`\tl_put_right:Nn` 498, 614, 659, 676, 729, 2316, 2347, 2394, 2404, 2417, 2432, 2438, 2443, 2467, 2472, 2479, 2482, 2492, 2497, 2500, 2506, 2897, 2900, 2907, 2909, 2936, 2941, 2946, 2949, 2958, 2971, 2974, 2980, 2985, 2995, 4729, 4733
`\tl_remove_all:Nn` 5071
`\tl_remove_once:Nn` 2382, 2921
`\tl_replace_all:Nnn` 502, 5106
`\tl_reverse:N` 2381, 2383, 2920, 2922
`\tl_set:Nn` . 58, 259, 269, 318, 319, 326, 327, 334, 335, 467, 541, 545, 550, 551, 603, 648, 720, 930, 944, 956, 968, 1851, 1952, 2216, 2226, 2247, 2255, 2536, 2747, 3014, 3059, 3072, 4735, 4758, 5069, 5105, 5175
`\tl_set_eq:NN` 508, 609, 612, 656, 658, 673, 675, 726, 728, 2380, 2919, 2932, 3265, 3269, 3996, 3998
`\tl_to_str:n` 1822, 1828, 1833, 4882
`\tl_trim_spaces:n` . . . 498, 5058, 5069, 5075, 5091
`\tl_use:N` 504, 507, 625, 690, 697, 740, 1001, 1005, 1009, 1013, 1017, 1021, 1025, 1029, 1033, 1037, 1041, 1045, 1049, 1053, 1057, 1061, 2370, 2387, 2395, 2406, 2419, 2424, 2435, 3100, 3106, 3134, 3161, 3162, 3169, 3256, 3260, 3268, 3295, 3296, 3302, 3419, 3605, 4003, 4275, 4330, 4532, 4539, 4564, 4567, 4805, 4812, 4817, 4823, 4828, 4937, 4938, 4939, 4940, 4941, 4959, 5054, 5173

token commands:

`\token_to_str:N` 450
topsep 799
`\topskip` 1199, 1371
`\typeout` 241, 242, 247, 248, 418, 421, 431, 432

U

`\u` 226, 2788
`\unkern` 249
unknown 3189, 3211, 3229
`\unskip` 243
use commands:
`\use:N` 233, 3166, 3421
`\use:n` 1702, 2259, 4888, 4980
`\use_none:nn` 442, 5112
`\usecounter` 3351, 3395

V

`\value` 1765, 1771, 1778, 1784, 1792, 1798, 1805, 1811
vbox commands:
`\vbox_set:Nn` 3972, 3977
`\vbox_set_top:Nn` 4273, 4328
`\vspace` . 860, 871, 1601, 1604, 1615, 1618, 1628, 1630, 1639, 1641, 1650, 1652, 1661, 1663, 1672, 1674, 1683, 1685, 3901, 3948

W

widest 773
wrap-ans 2151
wrap-label 511
wrap-label* 511
wrap-opt 2151

Z

`\z` 2788