

enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-07-15^{*}

©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 multicol	3	6.2.1	Keys for <code>\anskey</code>	12
1.3.4	Support for minipage	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 keyval 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	132
5.4	Keys for start, series and resume	9			
5.5	Keys for multicol	10			
5.6	Keys for minipage	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 multicol, vertical alignment`

^{*}This file describes a documentation for v1.0, last revised 2024-07-15.

[†]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

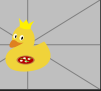
B)

A

C)

A

D)



E)

5. Question with image on left side:

A) value

B) value

C) value

D) correct

E) value

B
- Where what we are interested in the <label> and a “short note” that we leave as an explanation, and then print them:
- ©2024 by Pablo González L
- 2 / 146

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

enumext.pdf

README.md

enumext.dtx

»

»

»

»

TDS:tex/latex/enumext/

TDS:doc/latex/enumext/

TDS:doc/latex/enumext/

TDS:source/latex/enumext/

The package is loaded in the usual way:

\usepackage{enumext}

1.2 The concept of left margin

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

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



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

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



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

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

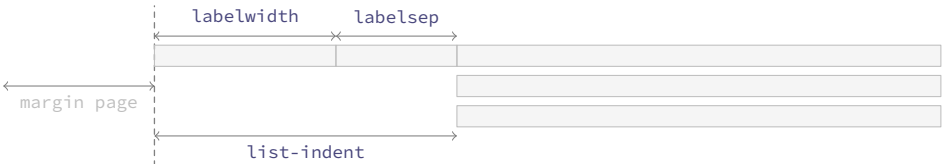


Figure 3: Default horizontal lengths `list-offset=0pt`, `list-indent=\labelwidth+\labelsep` in `enumext`.

1.3 User interface

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

1.3.1 Internal counters

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

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

1.3.2 Public dimension

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

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

1.3.3 Support for multicol

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



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

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

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

1.3.4 Support for minipage

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



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

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

1.3.5 The \label and \ref system

This package provides a user interface like the `enumitem`[6] package to customize the references which is activated by the `ref` key (§5.1), the standard \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.
- ★ 3. This text is in the first level.

2.3 The command \item*

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

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

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

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

2.3.1 Keys for `\item*`

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

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

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

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

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

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

3 The command `\setenumext`

<code>\setenumext</code>	<code>\setenumext{$\langle key = val \rangle$}</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={\langle store name \rangle}]
  \item Text \anskey{answer}
  \item Text
  \begin{keyans}
    ...
  \end{keyans}
\end{enumext}
```

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

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

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

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 = {\langle store name \rangle}` default: *not set*

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

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

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

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

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

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

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

6.1.1 Keys for label and ref

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

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

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

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

6.1.2 Keys for wrap and display

- `wrap-ans` = { $\langle code \rangle$ { $\#1$ } *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 { $\langle code \rangle$ } must be passed between braces and only affects the *argument* or *body* and NOT the “stored content” in the *sequence* and *prop list* { $\langle store name \rangle$ } set by `save-ans` key. If this key is passed using `\setenumext` it is necessary to use double ‘{ $\#1$ }’.
- `wrap-opt` = { $\langle code \rangle$ { $\#1$ } *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 { $\langle code \rangle$ } must be passed between braces and only affects the current *optional argument* and NOT the “stored content” in the *sequence* and *prop list* { $\langle store name \rangle$ } set by `save-ans` key. If this key is passed using `\setenumext` it is necessary to use double ‘{ $\#1$ }’.
- `show-ans` = { $\langle true | false \rangle$ } default: `false`
 Displays the *argument* passed to the `\anskey`, the *body* for `anskey*` environment, the $\langle label \rangle$ 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` = { $\langle symbol \rangle$ } 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` = { $\langle left | right \rangle$ } 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` = { $\langle true | false \rangle$ } default: `false`
 Displays the *position* occupied by the “stored content” by `\anskey`, `anskey*`, `\item*` and `\anspic*` in the *prop list* { $\langle store name \rangle$ } 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` = { $\langle true | false \rangle$ } 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` $\langle value forbidden \rangle$ default: `not used`
 This is a *meta-key* that does not receive an argument and disables the structure stored in the *sequence* { $\langle store name \rangle$ } 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* { $\langle store name \rangle$ }.

6.2 The command `\anskey`

`\anskey` $\langle \text{anskey}[\langle keys \rangle] \{ \langle content \rangle \} \rangle$

The command `\anskey` takes a mandatory non empty argument { $\langle content \rangle$ } and “stores” it in the *sequence* and *prop list* { $\langle store name \rangle$ } 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 { $\langle content \rangle$ } passed to `\anskey` when “storing” in the *sequence* { $\langle store name \rangle$ } has the form `\item \langle content \rangle`, the following $\langle keys \rangle$ allow modifying the way in which it is “stored” in the *sequence*.

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

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

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

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

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

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

Example

```

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

```



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

1. Text containing a question.

A) Choice
C) Choice
E) Choice

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

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


Some text

6.5 The environment keyanspic

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

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



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

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

6.5.1 The command \anspic

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

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

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

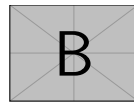
Example

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

1. Question with images.



A)



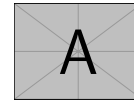
B)



C)



D)



* E)[note]

6.6 Printing stored content

6.6.1 The command `\getkeyans`

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

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

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

6.6.2 The command `\foreachkeyans`

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

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

Options for command

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

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

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

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

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

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

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

6.6.3 The command `\printkeyans`

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

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

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

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

established for the second, third and fourth levels plus the keys `nosep`, `first=\small`, `font=\small`. If the environment `enumext*` is saved within the *sequence* $\{\langle store\ name\rangle\}$ it will have the same default values plus the keys `nosep`, `first=\small`, `font=\small`.

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

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

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

Example

```
\begin{enumext}[save-ans=sample,columns=2,show-pos=true,nosep,save-ref=true]
  \item Factor  $3x+3y+3z$ . \anskey{$3(x+y+z)$}
  \item True False

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

  \item Related to Linux

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

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

```
\printkeyans{sample}
```

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

[1]

2. True False

(a)

[2]

3. Related to Linux

(a) You use linux?

[3]

(b) Rate the following package and class

i.

[4]

ii.

[5]

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

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

2. (a) Very True!

3. (a) Yes

(b) i. very good

ii. obsolete
- *

*

*

*

*

7 Full examples

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

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


- II. $\alpha = \delta$

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

A

I only

B

II only

C

I and II only

D

I and III only

E

I, II, and III
3. Third type of questions
- (1) $2\alpha + 2\delta = 90^\circ$

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

A

value

B

value

C

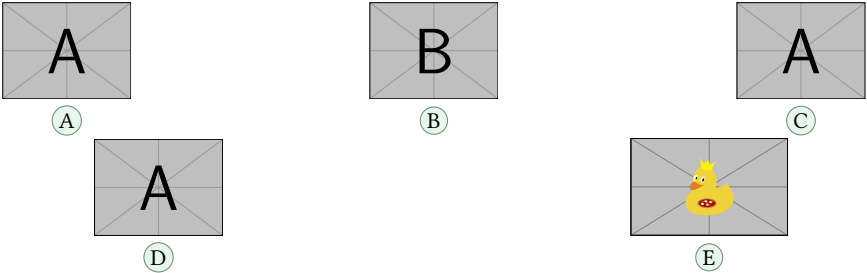
value

D

value

E

value
4. Question with image and label below:



5. Question with image on left side:
- A

value

B

value

C

value

D

correct

E

value

Test keys

1. B, $x = 5$

2. D

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

* 5. D, other note

*
- *

*

*

Example 4

A “simple worksheet” using ducks :) 🦆

Factor $x^2 - 2x + 1$

Factor $3x + 3y + 3z$

The following questions need to be cuaqtified :)

True False

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

Related to Linux

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

ii. `xsim`

iii. `exsheets`

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

1. $(x - 1)^2$

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

3. (a) False

(b) Very True!

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

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

* ii. very good

* iii. obsolete

*
- *

*

*

Example 5

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

- 1
- Which choice best describes what happens in the passage?
- A) One character argues with another character who intrudes on her home.
- B) One character receives a surprising request

- from another character.
- C) One character reminisces about choices she has made over the years.
- D) One character criticizes another character for pursuing an unexpected course of action.

2

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

- Something

A short one-line description.
This is an entry *without* a label.
- Something

A short one-line description.
- Something long

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

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

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

Final notes

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

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

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

Why list environments?

The answer is simple, first I love the beauty of its syntax and many of what I had already written used the `enumerate` environment or lists created using the `enumitem` package. In my mind I thought: how complicated could it be to write a package that looked like `enumitem`? It seemed simple enough, of course I didn’t have

in mind the mess I was getting into working with `list` environments, `minipage` and adding support for the `multicol` and `hyperref` packages.

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

Why not random questions and other utilities

The “*random*” type questions I love and hate them at the same time, although they simplify a lot the work when creating a multiple choice test, but you lose the beauty of typesetting a document with \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 2024-07-15 – First public release.

11 Index of Documentation

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

C	
Document class:	
article	2
book	2
exam	2
letter	2
report	2
\columnbreak	4, 12
\columnsep	10
Commands provide by enumext:	
\anskey	11–13
\anspic	11, 12, 15
\foreachkeyans	16
\getkeyans	12, 16
\item*	5–7, 11, 12, 14, 15
\item	5–10, 12, 14
\miniright	10
\printkeyans	6, 11, 16
\setenumextmeta	6
\setenumext	5–7, 11, 12, 14, 17
Counters defined by enumext:	
enumXiii	4
enumXii	4
enumXiv	4
enumXi	4
enumXviii	4
enumXvii	4
enumXvi	4
enumXv	4
E	
Environments provide by enumext:	
anskey*	11–13
enumext*	4–14, 16, 17
enumext	4–14, 16, 17, 20
keyans*	4–14
keyanspic	4, 7, 8, 11–13, 15, 20
keyans	4–9, 11–15, 20
Environments:	
Verbatim	13
enumerate	1, 3, 5, 21
figure	5
list	3, 9, 22
minipage	3–5, 10, 22
multicols	3, 4, 10
table	5
task	5
F	
\footnote	5
I	
\itemsep	8
K	
Keys for \anskey provide by enumext:	
break-col	12
item-join	12
item-pos*	13
item-star	12, 13
item-sym*	13
Keys for \foreachkeyans provide by enumext:	
after	16
before	16
sep	16
start	16
step	16
stop	16
wrapper	16
Keys for anskey* provide by enumext:	
break-col	12
force-eol	13
item-join	12
item-pos*	13
item-star	12, 13
item-sym*	13
overwrite	13
write-env	13
Keys for environments provide by enumext:	
above*	8
above	8
after	9, 10
align	7, 21
base-fix	8
before*	9
before	9
below*	9
below	8
check-ans	12
columns-sep	4, 10
columns	4, 8, 10
first	9
font	7
item-pos*	5, 6
item-sym*	5, 6
itemindent	9
itemsep	8, 15
labelsep	3–7, 9, 10, 12, 20, 21
labelwidth	3, 4, 6, 7, 9, 10, 12, 20, 21
labelwith	5
label	7, 9, 14, 20, 21
list-indent	3, 9
list-offset	3, 9, 21
listparindent	9
mark-ans	12
mark-pos	12
mark-ref	11
mini-env	4, 5, 8, 10
mini-right*	7, 10
mini-right	7, 10
mini-sep	4, 10
no-store	11–13
noitemsep	8
nosep	8, 20
overwrite	13
parsep	8, 15
partopsep	8
ref	4, 7
resume*	7, 10, 11
resume	7, 10, 11
rightmargin	9
save-ans	4, 6, 10–16

save-key	10, 11, 17	\linewidth	10
save-ref	4, 7, 11-13, 16	\listparindent	9
save-sep	11		
series	7, 10, 11	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, 22
wrap-ans	12	l3keys	7
wrap-label*	8, 21	l3prop	1, 21
wrap-label	7, 8	l3seq	1, 21
wrap-opt	12	multicol	1, 2, 4, 22
write-env	13	scontents	1, 2, 13
		task	5, 6
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   {2024-07-15}
7   {1.0}
8   {Enumerate exercise sheets}
```

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

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

12.4 Definition of variables

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

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

```

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

```

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

```

\c__enumext_counter_style_tl
\l__enumext_ref_key_arg_tl
\l__enumext_ref_the_count_tl
\l__enumext_the_counter_X_tl
\l__enumext_renew_the_count_X_tl

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

\l__enumext_current_widest_dim
\g__enumext_counter_styles_tl
\g__enumext_widest_label_tl
\l__enumext_label_width_by_box

```

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

```

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

```


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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```

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

```

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

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

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

The variables `\l__enumext_store_current_label_tl` and `\l__enumext_store_current_opt_arg_tl` save the *current label* and *optional argument* of `\item*` (§12.36) and `\anspic*` (§12.40.1) for the `keyans`, `keyans*` and `keyanspic` environments.

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

```

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

```

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

```

\l__enumext_setkey_tmpa_tl
\l__enumext_setkey_tmpb_tl
\l__enumext_setkey_tmpa_int
\l__enumext_setkey_tmpa_seq
\l__enumext_setkey_tmpb_seq

```

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

```

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

```

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

```

\l__enumext_meta_path_tl
\l__enumext_foreach_print_seq
\l__enumext_foreach_name_prop_tl
\l__enumext_foreach_default_keys_tl

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

\l__enumext_hyperref_bool
\l__enumext_footnotes_key_bool

```

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

```

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

```

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

```

\l__enumext_newlabel_arg_one_tl
\l__enumext_newlabel_arg_two_tl
\l__enumext_write_aux_file_tl
\l__enumext_label_copy_X_tl

```

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

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

```

154 \tl_new:N \l__enumext_newlabel_arg_one_tl
155 \tl_new:N \l__enumext_newlabel_arg_two_tl
156 \tl_new:N \l__enumext_write_aux_file_tl
157 \cs_set_protected:Npn \__enumext_tmp:n #1
158 {
159   \tl_new:c { \l__enumext_label_copy_#1_tl }
160 }
161 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }

```

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

```

\g__enumext_footnote_int
\g__enumext_footnote_arg_seq
\g__enumext_footnote_int_seq

```

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

```

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

```

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

```

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

```

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

```

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

```

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

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

```

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

```

(End of definition for `\c__enumext_all_envs_clist`.)

12.5 Some utility functions

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

`\seq_use:NV`

```

191 \cs_generate_variant:Nn \keys_precompile:nnN { neN }
192 \cs_generate_variant:Nn \seq_use:Nn { NV }

```

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

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

```

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

```

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

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

`__enumext_before_env:nn`

```

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

```

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

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

```

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

```

(End of definition for `__enumext_level:`.)

`__enumext_if_is_int:nT` A conditional function to know if the variable we are passing is an integer used by `start` and `widest` keys. This function is taken directly from the answer given by Henri Menke in [How to test if an expl3 function argument is an integer expression?](#)

`__enumext_if_is_int:nF`

`__enumext_if_is_int:nTF`

```

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

```

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

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

```

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

```

(End of definition for `__enumext_regex_counter_style:`.)

`__enumext_show_length:nnn`

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

```

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

(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.

```

229 \cs_new_protected:Npn \__enumext_unskip_unkern:
230 {
231     \int_case:nnT { \lastnodetype }
232     {
233         { 11 }
234         {
235             \typeout{SKIIIIIIIIIIIIIIIP}
236             \typeout{\the\lastskip}
237             \unskip
238         }
239         { 12 }
240         {
241             \typeout{KERRRRRRRRRRRRRRRRRN}
242             \typeout{\the\lastkern}
243             \unkern
244         }
245     }
246 }
```

(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:`

```

247 \cs_new_protected:Nn \__enumext_is_not_nested:
248 {
249     \str_case:en { \@currentenv }
250     {
251         {enumext}
252         {
253             \tl_set:Nn \l__enumext_envir_name_tl { enumext }
254             \bool_lazy_and:nnT
255             { \bool_not_p:n { \g__enumext_standar_bool } }
256             { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
257             {
258                 \bool_gset_true:N \g__enumext_standar_bool
259             }
260         }
261         {enumext*}
262         {
263             \tl_set:Nn \l__enumext_envir_name_tl { enumext* }
264             \bool_lazy_and:nnT
265             { \bool_not_p:n { \g__enumext_starred_bool } }
266             { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
267             {
268                 \bool_gset_true:N \g__enumext_starred_bool
269             }
270         }
271     }
272 }
```

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.

```

273 \cs_new_protected:Nn \__enumext_is_on_first_level:
274 {
275   \bool_lazy_all:nT
276   {
277     { \bool_if_p:N \g__enumext_standar_bool }
278     { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
279     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
280   }
281   {
282     \bool_set_true:N \l__enumext_standar_first_bool
283     \bool_set_true:N \l__enumext_anskey_env_bool
284     \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
285     \tl_gset:Ne \g__enumext_start_line_tl
286     {
287       on ~ line ~ \exp_not:V \inputlineno
288     }
289   }
290   \bool_lazy_all:nT
291   {
292     { \bool_if_p:N \g__enumext_starred_bool }
293     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
294     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
295   }
296   {
297     \bool_set_true:N \l__enumext_starred_first_bool
298     \bool_set_true:N \l__enumext_anskey_env_bool
299     \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
300     \tl_gset:Ne \g__enumext_start_line_tl
301     {
302       on ~ line ~ \exp_not:V \inputlineno
303     }
304   }
305 }

```

(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.

```

306 \cs_new_protected:Nn \__enumext_keyans_name_and_start:
307 {
308   \str_case:en { \@currenvir }
309   {
310     {keyans}
311     {
312       \tl_set:Nn \l__enumext_envir_name_tl { keyans }
313       \tl_set:Ne \l__enumext_check_start_line_env_tl
314       {
315         in ~ 'keyans' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
316       }
317     }
318     {keyans*}
319     {
320       \tl_set:Nn \l__enumext_envir_name_tl { keyans* }
321       \tl_set:Ne \l__enumext_check_start_line_env_tl
322       {
323         in ~ 'keyans*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
324       }
325     }
326     {keyanspic}
327     {
328       \tl_set:Nn \l__enumext_envir_name_tl { keyanspic }
329       \tl_set:Ne \l__enumext_check_start_line_env_tl
330       {
331         in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
332       }
333     }
334   }
335 }

```


(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:
336 \cs_new_protected:Nn \__enumext_reset_global_vars:
337 {
338   \__enumext_reset_global_int:
339   \__enumext_reset_global_bool:
340   \__enumext_reset_global_tl:
341 }
342 \cs_new_protected:Nn \__enumext_reset_global_int:
343 {
344   \int_gzero:N \g__enumext_item_number_int
345   \int_gzero:N \g__enumext_item_anskey_int
346   \int_gzero:N \g__enumext_item_answer_diff_int
347 }
348 \cs_new_protected:Nn \__enumext_reset_global_bool:
349 {
350   \bool_gset_false:N \g__enumext_check_ans_key_bool
351   \bool_gset_false:N \g__enumext_standar_bool
352   \bool_gset_false:N \g__enumext_starred_bool
353 }
354 \cs_new_protected:Nn \__enumext_reset_global_tl:
355 {
356   \tl_gclear:N \g__enumext_store_name_tl
357   \tl_gclear:N \g__enumext_start_line_tl
358   \tl_gclear:N \g__enumext_envir_name_tl
359 }
```

(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.

```
360 \cs_new_protected:Nn \__enumext_log_global_vars:
361 {
362   \msg_log:nneeee { enumext } { prop-seq-int-hook }
363   { \g__enumext_store_name_tl }
364   { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
365   { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
366   { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
367 }
```

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.

```
368 \cs_new_protected:Nn \__enumext_log_answer_vars:
369 {
370   \msg_log:nneee { enumext } { item-answer-hook }
371   { \int_use:N \g__enumext_item_number_int }
372   { \int_use:N \g__enumext_item_anskey_int }
373   { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int } }
374 }
```

(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.

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.

```
375 \__enumext_at_begin_document:n
376 {
```

```

377     \cs_new_eq:NN \__enumext_start_list:nn \list
378     \cs_new_eq:NN \__enumext_stop_list: \endlist
379     \cs_new_eq:NN \__enumext_item_std:w \item
380 }

```

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

```

381 \__enumext_at_begin_document:n
382 {
383     \cs_new_eq:NN \__enumext_minipage:w \minipage
384     \cs_new_eq:NN \__enumext_endminipage: \endminipage
385 }

```

(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)

```

386 \cs_new_protected:Nn \__enumext_internal_mini_page:
387 {
388     \int_compare:nNt { \l__enumext_level_int } = { 0 }
389     {
390         \DeclareDocumentEnvironment{__enumext_mini_page}{ m }
391         {
392             \__enumext_minipage:w [ t ] { ##1 }
393             \legacy_if_gset_false:n { @minipage }
394             \skip_vertical:N \c_zero_skip
395         }
396         {
397             \skip_vertical:N \c_zero_skip
398             \__enumext_endminipage:
399         }
400     }
401 }

```

(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.

```

402 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
403 \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”.

```

404 \cs_new_protected:Nn \__enumext_after_hyperref:
405 {
406     \IfPackageLoadedTF { hyperref }
407     {
408         \msg_info:nnn { enumext } { package-load } { hyperref }
409         \bool_set_true:N \l__enumext_hyperref_bool
410         \IfHyperBoolean{hyperfootnotes}
411         {

```

```

412         \typeout{hyperfootnotes=true}
413         \bool_set_true:N \l__enumext_footnotes_key_bool
414     }
415     { \typeout{hyperfootnotes=false} }
416 }
417 { }

```

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`.

```

418 \bool_if:NT \l__enumext_footnotes_key_bool
419 {
420     \IfPackageLoadedTF { footnotehyper }
421     {
422         \msg_info:nnn { enumext } { package-load } { footnotehyper }
423     }
424     {
425         \typeout{No ~ footnotehyper ~ load}
426         \typeout{Load ~ and ~ use ~ \string\makesavenoteenv{enumext*}}
427         \bool_set_false:N \l__enumext_footnotes_key_bool
428     }
429 }

```

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.

```

430 \bool_if:NTF \l__enumext_hyperref_bool
431 {
432     \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
433     \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
434 }
435 {
436     \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
437     \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
438 }
439 }

```

(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.

```

440 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
441 {
442     \protected@write \@auxout { }
443     {
444         \token_to_str:N \newlabel {#1}
445         {
446             {#2}
447             \bool_if:NT \l__enumext_hyperref_bool
448             { { \thepage } {#2} {#1} }
449             { }
450         }
451     }
452     \__enumext_hypertarget:nn {#1} { }
453     \__enumext_phantomsection:
454 }

```

(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.

```

455 \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.

```
456 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
457 {
458   \cs_if_exist:cTF { c@ #2 }
459   { \msg_fatal:nnn { enumext } { counters }{ #2 } }
460   {
461     \tl_set:Nn #1 { #2 }
462     \newcounter { #2 }
463   }
464 }
```

(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 465 \__enumext_define_counters:Nn \__enumext_counter_i_tl { enumXi }
enumXii 466 \__enumext_define_counters:Nn \__enumext_counter_ii_tl { enumXii }
enumXiii 467 \__enumext_define_counters:Nn \__enumext_counter_iii_tl { enumXiii }
enumXiv 468 \__enumext_define_counters:Nn \__enumext_counter_iv_tl { enumXiv }
enumXv 469 \__enumext_define_counters:Nn \__enumext_counter_v_tl { enumXv }
enumXvi 470 \__enumext_define_counters:Nn \__enumext_counter_vi_tl { enumXvi }
enumXvii 471 \__enumext_define_counters:Nn \__enumext_counter_vii_tl { enumXvii }
enumXviii 472 \__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.

```
473 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
474 {
475   \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
476   \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
477 }
478 \__enumext_register_counter_style:Nn \arabic { 0 }
479 \__enumext_register_counter_style:Nn \Alph { M }
480 \__enumext_register_counter_style:Nn \alph { m }
481 \__enumext_register_counter_style:Nn \Roman { VIII }
482 \__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.

```
483 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
484 {
485   \hbox_set:Nn \l__enumext_label_width_by_box {#2}
486   \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
487 }
488 \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.

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

```

490 {
491   \tl_clear_new:N #1
492   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
493   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
494   \tl_map_inline:Nn \g__enumext_counter_styles_tl
495   {
496     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
497     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
498     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
499   }
500   \__enumext_label_width_by_box:Nn \__enumext_current_widest_dim
501   { \tl_use:N \g__enumext_widest_label_tl }
502   \tl_set_eq:cN { the #2 } #1
503 }
504 \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.

```

505 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
506 {
507   \keys_define:nn { enumext / #1 }
508   {
509     font .tl_set:c = { l__enumext_label_font_style_#2_tl },
510     font .value_required:n = true,
511     labelsep .dim_set:c = { l__enumext_labelsep_#2_dim },
512     labelsep .initial:n = {0.3333em},
513     labelsep .value_required:n = true,
514     labelwidth .dim_set:c = { l__enumext_labelwidth_#2_dim },
515     labelwidth .value_required:n = true,
516     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
517     wrap-label .initial:n = {##1},
518     wrap-label .value_required:n = true,
519     wrap-label* .code:n = {
520       \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
521       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
522     },
523     wrap-label* .value_required:n = true,
524   }
525 }
526 \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.

```

527 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
528 {
529   \keys_define:nn { enumext / #1 }
530   {
531     align .choice:,
532     align / left .code:n =
533     {
534       \tl_clear:c { l__enumext_label_fill_left_#2_tl }
535       \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
536     },
537     align / right .code:n =
538     {
539       \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
540       \tl_clear:c { l__enumext_label_fill_right_#2_tl }
541     },
542     align / center .code:n =
543     {
544       \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
545       \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
546     },

```

```

547         align / unknown .code:n =
548             \msg_error:nneee { enumext } { unknown-choice }
549             { align } { left, ~ right, ~ center } { \exp_not:n {##1} },
550         align .initial:n = left,
551         align .value_required:n = true,
552     }
553 }
554 \clist_map_inline:nn
555 {
556     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
557 }
558 { \__enumext_tmp:nn #1 }

559 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
560 {
561     \keys_define:nn { enumext / #1 }
562     {
563         align .choice:,
564         align / left .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
565         align / right .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
566         align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
567         align / unknown .code:n =
568             \msg_error:nneee { enumext } { unknown-choice }
569             { align } { left, ~ right, ~ center } { \exp_not:n {##1} },
570         align .initial:n = left,
571         align .value_required:n = true,
572     }
573 }
574 \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

575 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
576 {
577     \keys_define:nn { enumext / #1 }
578     {
579         label .code:n = {
580             \__enumext_label_style:cnv { l__enumext_label_#2_tl }
581             { l__enumext_counter_#2_tl } {##1}
582             \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
583             \l__enumext_current_widest_dim
584         },
585         label .initial:n = #3,
586         label .value_required:n = true,
587         ref .code:n = \__enumext_standar_ref:n {##1},
588         ref .value_required:n = true,
589     }
590 }
591 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
592 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*) }
593 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
594 \__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.

```

595 \cs_new_protected:Npn \__enumext_standar_ref:n #1

```

```

596 {
597   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
598   \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
599   {
600     \msg_error:nnn { enumext } { key-ref-empty } { enumext }
601   }
602   {
603     \tl_set_eq:Nc
604       \l__enumext_ref_the_count_tl { \l__enumext_counter_ \__enumext_level: _tl }
605     \__enumext_regex_counter_style:
606     \tl_set_eq:Nc
607       \l__enumext_ref_the_count_tl { \l__enumext_the_counter_ \__enumext_level: _tl }
608     \tl_put_right:ce { \l__enumext_renew_the_count_ \__enumext_level: _tl }
609     {
610       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
611       { \exp_not:V \l__enumext_ref_key_arg_tl }
612     }
613   }
614 }

```

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

```

615 \cs_new_protected:Nn \__enumext_standar_ref:
616 {
617   \tl_if_empty:cF { \l__enumext_renew_the_count_ \__enumext_level: _tl }
618   {
619     \tl_use:c { \l__enumext_renew_the_count_ \__enumext_level: _tl }
620   }
621 }

```

(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
622 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
623 {
624   \keys_define:nn { enumext / #1 }
625   {
626     label .code:n = {
627       \__enumext_label_style:cvn { \l__enumext_label_#2_tl }
628       { \l__enumext_counter_#2_tl } {##1}
629       \dim_set_eq:cN { \l__enumext_labelwidth_#2_dim }
630       \l__enumext_current_widest_dim
631     },
632     label .initial:n = #3,
633     label .value_required:n = true,
634     ref .code:n = \__enumext_starred_ref:n {##1},
635     ref .value_required:n = true,
636   }
637 }
638 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
639 \__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`.

```

640 \cs_new_protected:Npn \__enumext_starred_ref:n #1
641 {
642   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
643   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
644   {
645     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
646     {
647       \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
648     }
649     {
650       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
651       \__enumext_regex_counter_style:
652       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
653       \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl

```



```

654         {
655             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
656             { \exp_not:V \l__enumext_ref_key_arg_tl }
657         }
658     }
659 }
660 \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
661 {
662     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
663     {
664         \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
665     }
666     {
667         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
668         \__enumext_regex_counter_style:
669         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
670         \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
671         {
672             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
673             { \exp_not:V \l__enumext_ref_key_arg_tl }
674         }
675     }
676 }
677 }

```

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.

```

678 \cs_new_protected:Nn \__enumext_starred_ref:
679 {
680     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
681     {
682         \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
683         {
684             \tl_use:N \l__enumext_renew_the_count_vii_tl
685         }
686     }
687     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
688     {
689         \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
690         {
691             \tl_use:N \l__enumext_renew_the_count_viii_tl
692         }
693     }
694 }

```

(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
695 \keys_define:nn { enumext / keyans }
696 {
697     label .code:n = {
698         \__enumext_label_style:cvn { \__enumext_label_v_tl }
699         { \__enumext_counter_v_tl } {#1}
700         \dim_set_eq:cN { \__enumext_labelwidth_v_dim }
701         \__enumext_current_widest_dim
702         \__enumext_label_style:cvn { \__enumext_label_vi_tl }
703         { \__enumext_counter_vi_tl } {#1}
704         \dim_set_eq:cN { \__enumext_labelwidth_v_dim }
705         \__enumext_current_widest_dim
706     },
707     label .initial:n = \Alph*,
708     label .value_required:n = true,
709     ref .code:n = \__enumext_keyans_ref:n {#1},
710     ref .value_required:n = true,
711 }

```

(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:`

```

712 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
713 {
714   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
715   \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
716   {
717     \msg_error:nnn { enumext } { key-ref-empty } { keyans }
718   }
719   {
720     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
721     \__enumext_regex_counter_style:
722     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
723     \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
724     {
725       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
726       { \exp_not:V \l__enumext_ref_key_arg_tl }
727     }
728   }
729 }

```

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

```

730 \cs_new_protected:Npn \__enumext_keyans_ref:
731 {
732   \tl_if_empty:NF \l__enumext_renew_the_count_v_tl
733   {
734     \tl_use:N \l__enumext_renew_the_count_v_tl
735   }
736 }

```

(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.

```

737 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
738 {
739   \__enumext_if_is_int:nTF { #3 }
740   {
741     \int_set:Nn #2 {#3}
742   }
743   {
744     \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
745     { \int_set:Nn #2 { \int_from_alph:n {#3} } }
746     \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
747     { \int_set:Nn #2 { \int_from_roman:n {#3} } }
748   }
749 }
750 \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.

```

751 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
752 {
753   \__enumext_if_is_int:nTF {#4}
754   {
755     \setcounter{enumX#1} { #4 }

```

```

756     }
757     {
758         \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
759         { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
760         \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
761         { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
762     }
763     \__enumext_label_width_by_box:cv
764     { l__enumext_labelwidth_#1_dim } { l__enumext_label_#1_tl }
765 }
766 \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.

```

767 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
768 {
769     \keys_define:nn { enumext / #1 }
770     {
771         start* .code:n = {
772             \__enumext_start_from:ccn
773             { l__enumext_label_#2_tl }
774             { l__enumext_start_#2_int } {##1}
775         },
776         start* .value_required:n = true,
777         start .code:n = {
778             \__enumext_start_from:cce
779             { l__enumext_label_#2_tl }
780             { l__enumext_start_#2_int } { \int_eval:n {##1} }
781         },
782         start .initial:n = 1,
783         start .value_required:n = true,
784         widest .code:n = {
785             \__enumext_widest_from:nccn {#2}
786             { l__enumext_label_#2_tl }
787             { l__enumext_labelwidth_#2_dim } {##1}
788         },
789         widest .value_required:n = true,
790     }
791 }
792 \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.

```

793 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
794 {
795     \keys_define:nn { enumext / #1 }
796     {
797         topsep .skip_set:c = { l__enumext_topsep_#2_skip },
798         topsep .initial:n = {#3},
799         topsep .value_required:n = true,
800         partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
801         partopsep .initial:n = {#4},
802         partopsep .value_required:n = true,
803         parsep .skip_set:c = { l__enumext_parsep_#2_skip },
804         parsep .initial:n = {#5},
805         parsep .value_required:n = true,
806         itemsep .skip_set:c = { l__enumext_itemsep_#2_skip },
807         itemsep .initial:n = {#6},
808         itemsep .value_required:n = true,
809         noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
810         noitemsep .value_forbidden:n = true,
811         nosepe .meta:n = {
812             itemsep = 0pt, parsep = 0pt,
813             topsep = 0pt, partopsep = 0pt,
814         },

```

```

815         nosep      .value_forbidden:n = true,
816     }
817 }

```

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

```

818 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
819 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
820 { 4.0pt plus 2.0pt minus 1.0pt }
821 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
822 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
823 { 2.0pt plus 1.0pt minus 1.0pt }
824 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
825 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
826 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
827 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
828 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
829 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
830 { 2.0pt plus 1.0pt minus 1.0pt }
831 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
832 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
833 { 4.0pt plus 2.0pt minus 1.0pt }
834 \__enumext_tmp:nnnnnn { keyans* } { viii } { 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 }

```

(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:
837 \cs_set_protected:Npn \__enumext_tmp:n #1
838 {
839     \keys_define:nn { enumext / #1 }
840     {
841         base-fix .bool_set:N = \l__enumext_base_line_fix_bool,
842         base-fix .initial:n = false,
843         base-fix .value_forbidden:n = true,
844     }
845 }
846 \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)

```

847 \cs_new_protected:Nn \__enumext_nested_base_line_fix:
848 {
849     \bool_lazy_and:nnT
850     { \bool_if_p:N \l__enumext_standar_first_bool }
851     { \bool_if_p:N \l__enumext_base_line_fix_bool }
852     {
853         \mode_leave_vertical:
854         \vspace { -\baselineskip }
855         \keys_set:nn { enumext / level-1 }
856         {
857             topsep = 0pt, above = 0pt, above* = 0pt,
858         }
859     }
860     \bool_lazy_and:nnT
861     { \bool_if_p:N \l__enumext_starred_first_bool }
862     { \bool_if_p:N \l__enumext_base_line_fix_bool }
863     {
864         \mode_leave_vertical:
865         \vspace { -\baselineskip }
866         \keys_set:nn { enumext / enumext* }
867         {
868             topsep = 0pt, above = 0pt, above* = 0pt,

```

```

869     }
870   }
871   \bool_set_false:N \__enumext_base_line_fix_bool
872 }

```

• 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.

```

873 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
874 {
875   \keys_define:nn { enumext / #1 }
876   {
877     itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
878     itemindent .value_required:n = true,
879     rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
880     rightmargin .value_required:n = true,
881     listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
882     listparindent .value_required:n = true,
883     list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
884     list-offset .value_required:n = true,
885     list-indent .code:n =
886       \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
887       \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
888     list-indent .value_required:n = true,
889   }
890 }
891 \clist_map_inline:nn
892 {
893   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
894 }
895 { \__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.

```

896 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
897 {
898   \keys_define:nn { enumext / #1 }
899   {
900     itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
901     itemindent .value_required:n = true,
902     rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
903     rightmargin .value_required:n = true,
904     listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
905     listparindent .value_required:n = true,
906     list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
907     list-offset .value_required:n = true,
908     list-indent .meta:n = { list-offset = ##1 },
909     list-indent .value_required:n = true,
910   }
911 }
912 \clist_map_inline:nn
913 {
914   {enumext*}{vii}, {keyans*}{viii}
915 }
916 { \__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.

```

917 \cs_set_protected:Nn \__enumext_fake_item:
918 {
919   \dim_compare:nNt
920   { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
921   >

```

```

922     { \c_zero_dim }
923   {
924     \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
925     {
926       \exp_not:N \mode_leave_vertical:
927       \exp_not:n { \skip_horizontal:n }
928       { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
929       \ignorespaces
930     }
931   }
932 }
933 \cs_set_protected:Nn \__enumext_keyans_fake_item:
934 {
935   \dim_compare:nNnT
936   { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
937   {
938     \tl_set:Nc \l__enumext_fake_item_indent_v_tl
939     {
940       \exp_not:N \mode_leave_vertical:
941       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
942     }
943   }
944 }
945 \cs_set_protected:Nn \__enumext_fake_item_vii:
946 {
947   \dim_compare:nNnT
948   { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
949   {
950     \tl_set:Nc \l__enumext_fake_item_indent_vii_tl
951     {
952       \exp_not:N \mode_leave_vertical:
953       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
954     }
955   }
956 }
957 \cs_set_protected:Nn \__enumext_fake_item_viii:
958 {
959   \dim_compare:nNnT
960   { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
961   {
962     \tl_set:Nc \l__enumext_fake_item_indent_viii_tl
963     {
964       \exp_not:N \mode_leave_vertical:
965       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
966     }
967   }
968 }

```

(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.

```

969 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
970 {
971   \keys_define:nn { enumext / #1 }
972   {
973     show-length .bool_set:c = { l__enumext_show_length_#2_bool },
974     show-length .initial:n = false,
975   }
976 }
977 \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.

```

978 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
979 {
980   \keys_define:nn { enumext / #1 }
981   {
982     before .tl_set:c = { l__enumext_before_no_starred_key_#2_tl },
983     before .value_required:n = true,
984     before* .tl_set:c = { l__enumext_before_starred_key_#2_tl },
985     before* .value_required:n = true,
986     after .tl_set:c = { l__enumext_after_stop_list_#2_tl },
987     after .value_required:n = true,
988     first .tl_set:c = { l__enumext_after_list_args_#2_tl },
989     first .value_required:n = true,
990   }
991 }
992 \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}`.

```

993 \cs_new_protected:Nn \__enumext_before_args_exec:
994 {
995   \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
996 }

```

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}`

```

997 \cs_new_protected:Nn \__enumext_before_keys_exec:
998 {
999   \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
1000 }

```

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

```

1001 \cs_new_protected:Nn \__enumext_after_stop_list:
1002 {
1003   \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
1004 }

```

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.`

```

1005 \cs_new_protected:Nn \__enumext_after_args_exec:
1006 {
1007   \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
1008 }

```

(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:
1009 \cs_new_protected:Nn \__enumext_before_args_exec_v:
1010 {
1011   \tl_use:N \l__enumext_before_starred_key_v_tl
1012 }
1013 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
1014 {
1015   \tl_use:N \l__enumext_before_no_starred_key_v_tl
1016 }
1017 \cs_new_protected:Nn \__enumext_after_stop_list_v:
1018 {
1019   \tl_use:N \l__enumext_after_stop_list_v_tl
1020 }
1021 \cs_new_protected:Nn \__enumext_after_args_exec_v:
1022 {
1023   \tl_use:N \l__enumext_after_list_args_v_tl
1024 }

```

(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.

```
1025 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
1026 {
1027   \tl_use:N \l__enumext_before_starred_key_vii_tl
1028 }
1029 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
1030 {
1031   \tl_use:N \l__enumext_before_starred_key_viii_tl
1032 }
1033 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
1034 {
1035   \tl_use:N \l__enumext_before_no_starred_key_vii_tl
1036 }
1037 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
1038 {
1039   \tl_use:N \l__enumext_before_no_starred_key_viii_tl
1040 }
1041 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
1042 {
1043   \tl_use:N \l__enumext_after_stop_list_vii_tl
1044 }
1045 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
1046 {
1047   \tl_use:N \l__enumext_after_stop_list_viii_tl
1048 }
1049 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
1050 {
1051   \tl_use:N \l__enumext_after_list_args_vii_tl
1052 }
1053 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
1054 {
1055   \tl_use:N \l__enumext_after_list_args_viii_tl
1056 }
```

(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.

```
1057 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1058 {
1059   \keys_define:nn { enumext / #1 }
1060   {
1061     mini-env .dim_set:c = { l__enumext_minipage_right_#2_dim },
1062     mini-env .value_required:n = true,
1063     mini-sep .dim_set:c = { l__enumext_minipage_hsep_#2_dim },
1064     mini-sep .initial:n = 0.3333em,
1065     mini-sep .value_required:n = true,
1066     columns-sep .dim_set:c = { l__enumext_columns_sep_#2_dim },
1067     columns-sep .value_required:n = true,
1068     columns .int_set:c = { l__enumext_columns_#2_int },
1069     columns .initial:n = 1,
1070     columns .value_required:n = true,
1071   }
1072 }
1073 \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.

```
1074 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1075 {
1076   \keys_define:nn { enumext / #1 }
1077   {
1078     mini-right .tl_gset:c = { g__enumext_miniright_code_#2_tl },
1079     mini-right .value_required:n = true,
1080     mini-right* .code:n = {
```

```

1081 \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
1082 \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1083 },
1084 mini-right* .value_required:n = true,
1085 }
1086 }
1087 \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*.

```

1088 \cs_new_protected:Nn \__enumext_multi_set_vskip:
1089 {
1090   \skip_set:cn { \l__enumext_multicol_above_ \__enumext_level: _skip }
1091   {
1092     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1093   }
1094   \skip_set:cn { \l__enumext_multicol_below_ \__enumext_level: _skip }
1095   {
1096     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1097   }
1098   \__enumext_add_pre_parsep:
1099 }

```

(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*.

```

1100 \cs_new_protected:Nn \__enumext_add_pre_parsep:
1101 {
1102   \int_case:nn { \l__enumext_level_int }
1103   {
1104     { 2 }{
1105       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1106       {
1107         \skip_add:Nn \l__enumext_multicol_above_ii_skip { \l__enumext_parsep_i_skip }
1108       }
1109     }
1110     { 3 }{
1111       \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1112       {

```

```

1113         \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_ii_skip
1114     }
1115 }
1116 { 4 }{
1117     \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1118     {
1119         \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip
1120     }
1121     }
1122 }
1123 }

```

(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*.

```

1124 \cs_new_protected:Nn \__enumext_multi_addvspace:
1125 {
1126     \__enumext_multi_set_vskip:
1127     \mode_if_vertical:T
1128     {
1129         \skip_add:cn { \l__enumext_multicols_above_ \__enumext_level: _skip }
1130         {
1131             \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1132         }
1133         \skip_add:cn { \l__enumext_multicols_below_ \__enumext_level: _skip }
1134         {
1135             \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1136         }
1137     }
1138     %%\__enumext_unskip_unkern:
1139     \par\nopagebreak
1140     \addvspace{ \skip_use:c { \l__enumext_multicols_above_ \__enumext_level: _skip } }
1141 }

```

(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:`

```

1142 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1143 {
1144     \skip_set:Nn \l__enumext_multicols_above_v_skip
1145     {
1146         \l__enumext_topsep_v_skip
1147     }
1148     \skip_set:Nn \l__enumext_multicols_below_v_skip
1149     {
1150         \l__enumext_topsep_v_skip
1151     }
1152 }
1153 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1154 {
1155     \__enumext_keyans_multi_set_vskip:
1156     \mode_if_vertical:T
1157     {
1158         \skip_add:Nn \l__enumext_multicols_above_v_skip
1159         {
1160             \skip_use:N \l__enumext_partopsep_v_skip
1161         }
1162         \skip_add:Nn \l__enumext_multicols_below_v_skip
1163         {
1164             \skip_use:N \l__enumext_partopsep_v_skip
1165         }
1166     }
1167     \__enumext_unskip_unkern:
1168     \par\nopagebreak
1169     \addvspace{ \l__enumext_multicols_above_v_skip }
1170 }

```

(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`.

```

1171 \cs_new_protected:Nn \__enumext_minipage_set_skip:
1172 {
1173   \skip_set:Nn \__enumext_minipage_right_skip
1174   {
1175     \skip_use:c { \__enumext_topsep_ \__enumext_level: _skip }
1176   }
1177   \mode_if_vertical:T
1178   {
1179     \skip_add:Nn \__enumext_minipage_right_skip
1180     {
1181       \skip_use:c { \__enumext_partopsep_ \__enumext_level: _skip }
1182     }
1183   }
1184   \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:`.

```

1185   \skip_set_eq:cN
1186   { \__enumext_multicols_above_ \__enumext_level: _skip } \__enumext_minipage_right_skip
1187   \skip_set_eq:cN
1188   { \__enumext_multicols_below_ \__enumext_level: _skip } \__enumext_minipage_right_skip
1189   \__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`.

```

1190   \int_compare:nNnT
1191   { \int_use:c { \__enumext_columns_ \__enumext_level: _int } } > { 1 }
1192   {
1193     \skip_zero:N \topskip
1194     \skip_set_eq:Nc \multicolsep { \__enumext_multicols_above_ \__enumext_level: _skip }
1195   }
1196 }

```

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*.

```

1197 \cs_new_protected:Nn \__enumext_minipage_add_space:
1198 {
1199   \__enumext_minipage_set_skip:
1200   \__enumext_unskip_unkern:
1201   \mode_if_vertical:TF
1202   {
1203     \nopagebreak\nointerlineskip
1204   }
1205   {
1206     \par\nopagebreak\nointerlineskip
1207     \skip_zero:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1208   }
1209   \int_compare:nNnTF
1210   { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1211   {
1212     \addvspace{ 0.445\box_ht:N \strutbox }
1213   }
1214   {
1215     \addvspace{ 0.250\box_ht:N \strutbox }
1216   }
1217 }

```

(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.

```

1218 \cs_new_protected:Nn \__enumext_pre_itemsep_skip:
1219 {
1220   \int_case:nn { \l__enumext_level_int }
1221   {
1222     { 2 }{
1223       \skip_if_eq:nnTF
1224       { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1225       {
1226         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1227         \skip_set:Nn \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1228       }
1229       {
1230         \dim_compare:nNnT
1231         { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1232         {
1233           \skip_sub:Nn
1234           \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1235           \skip_sub:Nn
1236           \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1237           \skip_add:Nn
1238           \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1239           \skip_add:Nn
1240           \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1241         }
1242         \dim_compare:nNnT
1243         { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1244         {
1245           \skip_set:Nn \l_tmpa_skip
1246           {
1247             \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1248           }
1249           \skip_sub:Nn
1250           \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1251           \skip_sub:Nn
1252           \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1253           \skip_add:Nn
1254           \l__enumext_minipage_after_skip
1255           { 0.150\box_ht:N \strutbox + \l_tmpa_skip }
1256           \skip_add:Nn
1257           \l__enumext_multicols_below_ii_skip
1258           { 0.350\box_ht:N \strutbox + \l_tmpa_skip }
1259         }
1260       }
1261     }
1262   }

```

```

1260         }
1261     }
1262     { 3 }{
1263         \skip_if_eq:nnTF
1264         { \l__enumext_itemsep_ii_skip } { \c_zero_skip }
1265         {
1266             \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1267             \skip_set:Nn \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1268         }
1269         {
1270             \dim_compare:nnNT
1271             { \l__enumext_itemsep_ii_skip } < { \l__enumext_minipage_after_skip }
1272             {
1273                 \skip_sub:Nn
1274                 \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1275                 \skip_sub:Nn
1276                 \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1277                 \skip_add:Nn
1278                 \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1279                 \skip_add:Nn
1280                 \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1281             }
1282             \dim_compare:nnNT
1283             { \l__enumext_itemsep_ii_skip } > { \l__enumext_minipage_after_skip }
1284             {
1285                 \skip_set:Nn \l_tmpa_skip
1286                 {
1287                     \l__enumext_itemsep_ii_skip - \l__enumext_minipage_after_skip
1288                 }
1289                 \skip_sub:Nn
1290                 \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1291                 \skip_sub:Nn
1292                 \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1293                 \skip_add:Nn
1294                 \l__enumext_minipage_after_skip
1295                 { 0.150\box_ht:N \strutbox + \l_tmpa_skip }
1296                 \skip_add:Nn
1297                 \l__enumext_multicols_below_iii_skip
1298                 { 0.350\box_ht:N \strutbox + \l_tmpa_skip }
1299             }
1300         }
1301     }
1302     { 4 }{
1303         \skip_if_eq:nnTF { \l__enumext_itemsep_iii_skip } { \c_zero_skip }
1304         {
1305             \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1306             \skip_set:Nn \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1307         }
1308         {
1309             \dim_compare:nnNT
1310             { \l__enumext_itemsep_iii_skip } < { \l__enumext_minipage_after_skip }
1311             {
1312                 \skip_sub:Nn
1313                 \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1314                 \skip_sub:Nn
1315                 \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1316                 \skip_add:Nn
1317                 \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1318                 \skip_add:Nn
1319                 \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1320             }
1321             \dim_compare:nnNT
1322             { \l__enumext_itemsep_iii_skip } > { \l__enumext_minipage_after_skip }
1323             {
1324                 \skip_set:Nn \l_tmpa_skip
1325                 {
1326                     \l__enumext_itemsep_iii_skip - \l__enumext_minipage_after_skip
1327                 }
1328                 \skip_sub:Nn
1329                 \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1330                 \skip_sub:Nn

```



```

1331         \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1332     \skip_add:Nn
1333     \l__enumext_minipage_after_skip
1334     { 0.150\box_ht:N \strutbox + \l_tmpa_skip }
1335     \skip_add:Nn
1336     \l__enumext_multicols_below_iv_skip
1337     { 0.350\box_ht:N \strutbox + \l_tmpa_skip }
1338 }
1339 }
1340 }
1341 }
1342 }

```

(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`.

```

1343 \cs_new_protected:Nn \__enumext_keyans_minipage_set_skip:
1344 {
1345     \skip_zero:N \l__enumext_minipage_after_skip
1346     \skip_zero:N \l__enumext_minipage_left_skip
1347     \skip_zero:N \l__enumext_minipage_right_skip
1348     \skip_set:Nn \l__enumext_minipage_right_skip
1349     {
1350         \l__enumext_topsep_v_skip
1351     }
1352     \mode_if_vertical:T
1353     {
1354         \skip_add:Nn \l__enumext_minipage_right_skip
1355         {
1356             \l__enumext_partopsep_v_skip
1357         }
1358     }
1359     \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip
1360     \skip_set_eq:NN \l__enumext_multicols_above_v_skip \l__enumext_minipage_right_skip
1361     \skip_set_eq:NN \l__enumext_multicols_below_v_skip \l__enumext_minipage_right_skip
1362     \__enumext_keyans_pre_itemsep_skip:
1363     \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
1364     {
1365         \skip_zero:N \topskip
1366         \skip_set_eq:NN \multicolsep \l__enumext_minipage_right_skip
1367     }
1368 }
1369 \cs_new_protected:Nn \__enumext_keyans_minipage_add_space:
1370 {
1371     \__enumext_keyans_minipage_set_skip:
1372     \__enumext_unskip_unkern:
1373     \mode_if_vertical:TF
1374     {
1375         \nopagebreak\nointerlineskip
1376     }
1377     {
1378         \par\nopagebreak\nointerlineskip
1379         \skip_zero:N \l__enumext_partopsep_v_skip
1380     }
1381     \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1382     {
1383         \addvspace{ 0.445\box_ht:N \strutbox }
1384     }
1385     {
1386         \addvspace{ 0.250\box_ht:N \strutbox }
1387     }
1388 }
1389 \cs_new_protected:Nn \__enumext_keyans_pre_itemsep_skip:
1390 {
1391     \skip_if_eq:nnTF
1392     { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1393     {
1394         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }

```

```

1395     \skip_set:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1396   }
1397   {
1398     \dim_compare:nNnT
1399       { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1400     {
1401       \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1402       \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1403       \skip_add:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1404       \skip_add:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1405     }
1406     \dim_compare:nNnT
1407       { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1408     {
1409       \skip_set:Nn \l_tmpa_skip
1410       {
1411         \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1412       }
1413       \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1414       \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1415       \skip_add:Nn \l__enumext_minipage_after_skip
1416         { 0.150\box_ht:N \strutbox + \l_tmpa_skip }
1417       \skip_add:Nn \l__enumext_multicols_below_v_skip
1418         { 0.350\box_ht:N \strutbox + \l_tmpa_skip }
1419     }
1420   }
1421 }

```

(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*`.

```

1422 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1423 {
1424   \skip_zero_new:N \l__enumext_minipage_left_skip
1425   \skip_gzero_new:N \g__enumext_minipage_right_skip
1426   \skip_gzero_new:N \g__enumext_minipage_after_skip
1427   \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1428   {
1429     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1430     \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1431   }
1432   {
1433     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1434     \skip_gset:Nn \g__enumext_minipage_right_skip
1435       {
1436         \l__enumext_topsep_vii_skip
1437       }
1438     \skip_gset:Nn \g__enumext_minipage_after_skip
1439       {
1440         0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1441       }
1442   }
1443 }
1444 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1445 {
1446   \skip_zero_new:N \l__enumext_minipage_after_skip
1447   \skip_zero_new:N \l__enumext_minipage_left_skip
1448   \skip_zero_new:N \l__enumext_minipage_right_skip
1449   \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1450   {
1451     \skip_set:Nn \l__enumext_minipage_left_skip
1452       {
1453         0.5\box_dp:N \strutbox
1454       }
1455     \skip_set:Nn \l__enumext_minipage_right_skip
1456       {
1457         \l__enumext_partopsep_viii_skip

```

```

1458     }
1459     \skip_set:Nn \l__enumext_minipage_after_skip
1460     {
1461         1.6\box_dp:N \strutbox
1462     }
1463 }
1464 {
1465     \skip_set:Nn \l__enumext_minipage_left_skip
1466     {
1467         0.5875\box_dp:N \strutbox
1468     }
1469     \skip_set:Nn \l__enumext_minipage_right_skip
1470     {
1471         \l__enumext_topsep_viii_skip
1472     }
1473     \skip_set:Nn \l__enumext_minipage_after_skip
1474     {
1475         0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1476     }
1477 }
1478 }

```

(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.

```

1479 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1480 {
1481     \__enumext_mini_set_vskip_vii:
1482     \par\nopagebreak
1483     \addvspace { \l__enumext_minipage_left_skip }
1484 }
1485 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1486 {
1487     \__enumext_mini_set_vskip_viii:
1488     \par\nopagebreak
1489     \addvspace { \l__enumext_minipage_left_skip }
1490 }

```

(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.

```

1491 \NewDocumentCommand \miniright { s }
1492 {
1493     \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
1494     {
1495         \msg_error:nnn { enumext } { wrong-miniright-place }
1496     }
1497     % outside
1498     \bool_lazy_and:nnT
1499     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
1500     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
1501     {
1502         \msg_error:nnn { enumext } { wrong-miniright-place }
1503     }
1504     % starred env
1505     \bool_if:NT \l__enumext_starred_bool
1506     {

```

```

1507     \msg_error:nnn { enumext } { wrong-miniright-starred }
1508   }
1509   \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
1510   {
1511     \__enumext_keyans_mini_right_cmd:n {#1}
1512   }
1513   { \__enumext_mini_right_cmd:n {#1} }
1514 }

```

(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.

```

1515 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1516 {
1517   \dim_compare:nNnTF
1518   { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1519   {
1520     \__enumext_multicols_stop:
1521     \int_compare:nNnTF
1522     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } = { 1 }
1523     {
1524       \par\addvspace{ \l__enumext_minipage_after_skip }
1525     }
1526     \end__enumext_mini_page
1527     \hfill
1528     \__enumext_mini_page{ \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } }
1529     \par\nointerlineskip
1530     \addvspace { \l__enumext_minipage_right_skip }
1531     \bool_if:nF {#1}
1532     {
1533       \centering
1534     }
1535     \int_gzero:N \g__enumext_minipage_stat_int
1536   }
1537   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1538   % paranoia
1539   \RenewDocumentCommand \miniright { s }
1540   {
1541     \msg_error:nn { enumext } { many-miniright-used }
1542   }
1543 }

```

(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.

```

1544 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1545 {
1546   \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1547   {
1548     \__enumext_keyans_multicols_stop:
1549     \int_compare:nNnTF { \l__enumext_columns_v_int } = { 1 }
1550     {
1551       \par\addvspace{ \l__enumext_minipage_after_skip }
1552     }
1553     \end__enumext_mini_page
1554     \hfill
1555     \__enumext_mini_page{ \l__enumext_minipage_right_v_dim }
1556     \par\nointerlineskip
1557     \addvspace { \l__enumext_minipage_right_skip }
1558     \bool_if:nF {#1}
1559     {
1560       \centering
1561     }
1562     \int_gzero:N \g__enumext_minipage_stat_int

```

```

1563     }
1564     { \msg_error:nnn { enumext } { wrong-miniright-use } }
1565 % paranoia
1566 \RenewDocumentCommand \miniright { s }
1567 {
1568     \msg_error:nn { enumext } { many-miniright-used }
1569 }
1570 }

```

(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*
below
below*
1571 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1572 {
1573     \keys_define:nn { enumext / #1 }
1574     {
1575         above .skip_set:c = { \__enumext_vspace_above_#2_skip },
1576         above .value_required:n = true,
1577         above* .code:n = \bool_set_true:c { \__enumext_vspace_a_star_#2_bool }
1578             \keys_set:nn { enumext / #1 } { above = {##1} },
1579         above* .value_required:n = true,
1580         below .skip_set:c = { \__enumext_vspace_below_#2_skip },
1581         below .value_required:n = true,
1582         below* .code:n = \bool_set_true:c { \__enumext_vspace_b_star_#2_bool }
1583             \keys_set:nn { enumext / #1 } { below = {##1} },
1584         below* .value_required:n = true,
1585     }
1586 }
1587 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

12.23.1 Functions for above and below keys in enumext

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

```

1588 \cs_new_protected:Nn \__enumext_vspace_above:
1589 {
1590     \skip_if_eq:nnF
1591     { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1592     {
1593         \bool_if:cTF { \__enumext_vspace_a_star_ \__enumext_level: _bool }
1594         {
1595             \vspace*{ \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1596         }
1597         {
1598             \vspace { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1599         }
1600     }
1601 }

```

(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.

```

1602 \cs_new_protected:Nn \__enumext_vspace_below:
1603 {
1604     \skip_if_eq:nnF
1605     { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1606     {
1607         \bool_if:cTF { \__enumext_vspace_b_star_ \__enumext_level: _bool }
1608         {
1609             \vspace*{ \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1610         }

```

```

1611         {
1612             \vspace { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1613         }
1614     }
1615 }

```

(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.

```

1616 \cs_new_protected:Nn \__enumext_vspace_above_v:
1617 {
1618     \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1619     {
1620         \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1621         {
1622             \vspace*{ \l__enumext_vspace_above_v_skip }
1623         }
1624         { \vspace { \l__enumext_vspace_above_v_skip } }
1625     }
1626 }

```

(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.

```

1627 \cs_new_protected:Nn \__enumext_vspace_below_v:
1628 {
1629     \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1630     {
1631         \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1632         {
1633             \vspace*{ \l__enumext_vspace_below_v_skip }
1634         }
1635         { \vspace { \l__enumext_vspace_below_v_skip } }
1636     }
1637 }

```

(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:

```

1638 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1639 {
1640     \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1641     {
1642         \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1643         {
1644             \vspace*{ \l__enumext_vspace_above_vii_skip }
1645         }
1646         { \vspace { \l__enumext_vspace_above_vii_skip } }
1647     }
1648 }
1649 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1650 {
1651     \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1652     {
1653         \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1654         {
1655             \vspace*{ \l__enumext_vspace_above_viii_skip }
1656         }
1657         { \vspace { \l__enumext_vspace_above_viii_skip } }
1658     }
1659 }

```

(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:`

```

1660 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1661 {
1662   \skip_if_eq:nnF { \__enumext_vspace_below_vii_skip } { \c_zero_skip }
1663   {
1664     \bool_if:NTF \__enumext_vspace_b_star_vii_bool
1665     {
1666       \vspace*{ \__enumext_vspace_below_vii_skip }
1667     }
1668     { \vspace { \__enumext_vspace_below_vii_skip } }
1669   }
1670 }
1671 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1672 {
1673   \skip_if_eq:nnF { \__enumext_vspace_below_viii_skip } { \c_zero_skip }
1674   {
1675     \bool_if:NTF \__enumext_vspace_b_star_viii_bool
1676     {
1677       \vspace*{ \__enumext_vspace_below_viii_skip }
1678     }
1679     { \vspace { \__enumext_vspace_below_viii_skip } }
1680   }
1681 }

```

(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*`.

```

1682 \cs_set_protected:Npn \__enumext_tmp:n #1
1683 {
1684   \keys_define:nn { enumext / #1 }
1685   {
1686     series .str_set:N = \__enumext_series_str,
1687     series .value_required:n = true,
1688     resume .code:n = \__enumext_resume_series:n {##1},
1689     resume* .code:n = \__enumext_resume_starred:,
1690     resume* .value_forbidden:n = true,
1691   }
1692 }
1693 \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`

```

1694 \cs_new:Npn \__enumext_filter_series:n #1
1695 {
1696   \use:e
1697   {
1698     \keyval_parse:NNn
1699     \__enumext_filter_series_key:n
1700     \__enumext_filter_series_pair:nn {#1}
1701   }
1702 }

```

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.

```

1703 \cs_new:Npn \__enumext_filter_series_key:n #1
1704 {
1705   \str_case:nnF {#1}
1706   {
1707     { resume } {} { resume* } {} { base-fix } {}
1708   }
1709   { , { \exp_not:n {#1} } }
1710 }

```

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.

```

1711 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1712 {
1713   \str_case:nnF {#1}
1714   {
1715     { series } {} { resume } {} { start } {}
1716     { start* } {} { save-ans } {} { save-key } {}
1717   }
1718   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1719 }

```

(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).

```

1720 \cs_new_protected:Npn \__enumext_parse_series:n #1
1721 {
1722   \str_if_empty:NTF \l__enumext_series_str
1723   {
1724     \bool_if:NF \l__enumext_resume_active_bool
1725     {
1726       \__enumext_resume_last:n {#1}
1727     }
1728   }
1729   {
1730     \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str_tl }
1731     \tl_gset:ce { g__enumext_series_ \l__enumext_series_str_tl }
1732     { \__enumext_filter_series:n {#1} }
1733     \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str_int }
1734     {
1735       \int_new:c { g__enumext_series_ \l__enumext_series_str_int }
1736     }
1737   }
1738 }

```

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.

```

1739 \cs_new_protected:Npn \__enumext_resume_last:n #1
1740 {
1741   \bool_if:NT \l__enumext_standar_first_bool
1742   {
1743     \tl_gclear:N \g__enumext_standar_series_tl
1744     \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1745   }
1746   \bool_if:NT \l__enumext_starred_first_bool
1747   {
1748     \tl_gclear:N \g__enumext_starred_series_tl
1749     \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1750   }
1751 }

```

(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).

```

1752 \cs_new_protected:Nn \__enumext_resume_save_counter:
1753 {
1754   \bool_if:NT \g__enumext_standar_bool
1755   {
1756     \tl_if_empty:NF \l__enumext_series_str
1757     {
1758       \int_gset_eq:cN
1759       { g__enumext_series_ \l__enumext_series_str_int } \value{enumXi}
1760     }
1761     \tl_if_empty:NTF \l__enumext_resume_name_tl
1762     {
1763       \str_if_empty:NT \l__enumext_series_str
1764       {
1765         \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1766       }
1767     }
1768     {
1769       \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl_int }
1770       {
1771         \int_gset_eq:cN
1772         { g__enumext_series_ \l__enumext_resume_name_tl_int } \value{enumXi}
1773       }
1774     }
1775     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl_int }
1776     {
1777       \int_gset_eq:cN
1778       { g__enumext_resume_ \l__enumext_store_name_tl_int } \value{enumXi}
1779     }
1780   }
1781   \bool_if:NT \g__enumext_starred_bool
1782   {
1783     \tl_if_empty:NF \l__enumext_series_str
1784     {
1785       \int_gset_eq:cN
1786       { g__enumext_series_ \l__enumext_series_str_int } \value{enumXvii}
1787     }
1788     \tl_if_empty:NTF \l__enumext_resume_name_tl
1789     {
1790       \str_if_empty:NT \l__enumext_series_str
1791       {
1792         \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1793       }
1794     }
1795     {
1796       \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl_int }
1797       {
1798         \int_gset_eq:cN
1799         { g__enumext_series_ \l__enumext_resume_name_tl_int } \value{enumXvii}
1800       }
1801     }
1802     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl_int }
1803     {
1804       \int_gset_eq:cN
1805       { g__enumext_resume_ \l__enumext_store_name_tl_int } \value{enumXvii}
1806     }
1807   }
1808 }

```

(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.

```

1809 \cs_new_protected:Npn \__enumext_resume_series:n #1
1810 {
1811   \tl_if_empty:NTF {#1}
1812   {
1813     \__enumext_resume_counter:n { }
1814   }
1815   {
1816     \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1817     {
1818       \__enumext_resume_counter:n {#1}
1819       \bool_if:NT \g__enumext_standar_bool
1820       {
1821         \keys_set:nv { enumext / level-1 }
1822         { g__enumext_series_ \tl_to_str:n {#1} _tl }
1823       }
1824       \bool_if:NT \g__enumext_starred_bool
1825       {
1826         \keys_set:nv { enumext / enumext* }
1827         { g__enumext_series_ \tl_to_str:n {#1} _tl }
1828       }
1829     }
1830     {
1831       \bool_if:NT \g__enumext_standar_bool
1832       {
1833         \msg_error:nnn { enumext } { unknown-series } {#1}
1834       }
1835       \bool_if:NT \g__enumext_starred_bool
1836       {
1837         \msg_error:nnn { enumext } { unknown-series } {#1}
1838       }
1839     }
1840   }
1841 }

```

(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`.

```

1842 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1843 {
1844   \bool_set_true:N \l__enumext_resume_active_bool
1845   \tl_set:Nn \l__enumext_resume_name_tl {#1}
1846   \tl_if_empty:NTF \l__enumext_resume_name_tl
1847   {
1848     \__enumext_resume_counter:
1849   }
1850   {
1851     \__enumext_resume_counter_series:
1852   }
1853   \__enumext_resume_counter_save_ans:
1854 }

```

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.

```

1855 \cs_new_protected:Nn \__enumext_resume_counter:
1856 {
1857   \bool_if:NT \g__enumext_standar_bool

```

```

1858     {
1859         \int_gincr:N \g__enumext_resume_int
1860         \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1861     }
1862     \bool_if:NT \g__enumext_starred_bool
1863     {
1864         \int_gincr:N \g__enumext_resume_vii_int
1865         \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1866     }
1867 }

```

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.

```

1868 \cs_new_protected:Nn \__enumext_resume_counter_series:
1869 {
1870     \bool_if:NT \g__enumext_standar_bool
1871     {
1872         \int_set:Nn \l__enumext_start_i_int
1873         {
1874             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1875         }
1876     }
1877     \bool_if:NT \g__enumext_starred_bool
1878     {
1879         \int_set:Nn \l__enumext_start_vii_int
1880         {
1881             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1882         }
1883     }
1884 }

```

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.

```

1885 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1886 {
1887     \bool_lazy_and:nnT
1888     { \bool_if_p:N \l__enumext_standar_first_bool }
1889     { \bool_if_p:N \l__enumext_store_active_bool }
1890     {
1891         \int_set:Nn \l__enumext_start_i_int
1892         {
1893             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1894         }
1895     }
1896     \bool_lazy_and:nnT
1897     { \bool_if_p:N \l__enumext_starred_first_bool }
1898     { \bool_if_p:N \l__enumext_store_active_bool }
1899     {
1900         \int_set:Nn \l__enumext_start_vii_int
1901         {
1902             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1903         }
1904     }
1905 }

```

(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.

```

1906 \cs_new_protected:Nn \__enumext_resume_starred:
1907 {
1908     \bool_if:NT \g__enumext_standar_bool
1909     {
1910         \tl_if_empty:NF \g__enumext_standar_series_tl
1911         {
1912             \__enumext_resume_counter:n { }

```

```

1913         \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1914     }
1915 }
1916 \bool_if:NT \g__enumext_starred_bool
1917 {
1918     \tl_if_empty:NF \g__enumext_starred_series_tl
1919     {
1920         \__enumext_resume_counter:n { }
1921         \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1922     }
1923 }
1924 }

```

(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*`.

```

1925 \cs_set_protected:Npn \__enumext_tmp:n #1
1926 {
1927     \keys_define:nn { enumext / #1 }
1928     {
1929         save-ans .code:n = \__enumext_storing_set:n {##1},
1930         save-ans .value_required:n = true,
1931     }
1932 }
1933 \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:` 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:`.

```

1934 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
1935 {
1936     \msg_term:nnVV { enumext } { save-ans-log }
1937     \g__enumext_envir_name_tl \l__enumext_store_name_tl
1938 }
1939 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
1940 {
1941     \msg_term:nnVV { enumext } { save-ans-log-hook }
1942     \g__enumext_envir_name_tl \g__enumext_store_name_tl
1943 }

```

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

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

```

1944 \cs_new_protected:Npn \__enumext_storing_set:n #1
1945 {
1946     \tl_set:Ne \l__enumext_store_name_tl {#1}
1947     \tl_if_empty:NTF \l__enumext_store_name_tl
1948     {
1949         \bool_lazy_or:nnT
1950         { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1951         {
1952             \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
1953         }
1954     }
1955     {
1956         \bool_lazy_or:nnT

```

```

1957         { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1958     {
1959         __enumext_start_save_ans_msg:
1960         __enumext_storing_exec:
1961     }
1962 }
1963 }

```

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*`.

```

1964 \cs_new_protected:Nn \__enumext_storing_exec:
1965 {
1966     \bool_set_true:N \l__enumext_store_active_bool
1967     \bool_set_true:N \l__enumext_check_answers_bool
1968     \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1969     \__enumext_anskey_env_make:V \l__enumext_store_name_tl
1970     \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1971     {
1972         \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
1973         \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1974     }
1975     \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1976     {
1977         \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
1978         \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1979     }
1980     \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1981     {
1982         \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
1983         \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1984     }
1985 }

```

(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 1986 \cs_set_protected:Npn \__enumext_tmp:n #1
no-store 1987 {
1988   \keys_define:nn { enumext / #1 }
1989   {
1990     check-ans .bool_set:N = \__enumext_check_ans_key_bool,
1991     check-ans .initial:n = false,
1992     check-ans .value_required:n = true,
1993     no-store .code:n = {
1994       \bool_set_false:N \__enumext_check_answers_bool
1995       \bool_set_false:N \__enumext_check_ans_key_bool
1996     },
1997     no-store .value_forbidden:n = true,
1998   }
1999 }
2000 \clist_map_inline:nn
2001 {
2002   level-1, level-2, level-3, level-4, enumext*
2003 }
2004 { \__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.

```

2005 \cs_new_protected:Nn \__enumext_check_ans_active:
2006 {
2007   \tl_if_empty:NF \__enumext_store_name_tl
2008   {
2009     \bool_if:NT \__enumext_check_answers_bool
2010     {
2011       \__enumext_check_ans_level:
2012     }
2013   }
2014 }
```

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*”.

```

2015 \cs_new_protected:Nn \__enumext_check_ans_level:
2016 {
2017   \int_case:nn { \__enumext_level_int }
2018   {
2019     { 1 }{
2020       \bool_lazy_all:nT
2021       {
2022         { \bool_if_p:N \g__enumext_starred_bool }
2023         { \int_compare_p:nNn { \__enumext_level_h_int } = { 1 } }
2024       }
2025       {
2026         \int_gdecr:N \g__enumext_item_number_int
2027         \bool_set_false:N \__enumext_item_number_bool
2028       }
2029     }
2030     { 2 }{
2031       \int_gdecr:N \g__enumext_item_number_int
2032       \bool_set_false:N \__enumext_item_number_bool
2033     }
2034     { 3 }{
2035       \int_gdecr:N \g__enumext_item_number_int
2036       \bool_set_false:N \__enumext_item_number_bool
2037     }
2038     { 4 }{
2039       \int_gdecr:N \g__enumext_item_number_int
2040       \bool_set_false:N \__enumext_item_number_bool

```

```

2041         }
2042     }

```

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.

```

2043     \int_case:nn { \l__enumext_level_h_int }
2044     {
2045         { 1 }{
2046             \bool_lazy_all:nT
2047             {
2048                 { \bool_if_p:N \g__enumext_standar_bool }
2049                 { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
2050             }
2051             {
2052                 \int_gdecr:N \g__enumext_item_number_int
2053                 \bool_set_false:N \l__enumext_item_number_bool
2054             }
2055         }
2056     }
2057 }

```

(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.

```

2058 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
2059 {
2060     \bool_lazy_and:nnT
2061     { \bool_if_p:N \l__enumext_check_ans_key_bool }
2062     { \bool_if_p:N \g__enumext_standar_bool }
2063     {
2064         \bool_gset_true:N \g__enumext_check_ans_key_bool
2065     }
2066     \bool_lazy_and:nnT
2067     { \bool_if_p:N \l__enumext_check_ans_key_bool }
2068     { \bool_if_p:N \g__enumext_starred_bool }
2069     {
2070         \bool_gset_true:N \g__enumext_check_ans_key_bool
2071     }
2072 }

```

(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:`.

```

2073 \cs_new_protected:Nn \__enumext_item_answer_diff:
2074 {
2075     \int_gset:Nn \g__enumext_item_answer_diff_int
2076     {
2077         \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
2078     }
2079 }

```

(End of definition for `__enumext_item_answer_diff:`.)

`__enumext_check_ans_show:`

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

```

2080 \cs_new_protected:Nn \__enumext_check_ans_show:
2081 {
2082     \int_case:nn { \g__enumext_item_answer_diff_int }
2083     {
2084         { -1 }{ \__enumext_check_ans_msg_less: }
2085         { 0 }{ \__enumext_check_ans_msg_same_ok: }
2086         { 1 }{ \__enumext_check_ans_msg_greater: }
2087     }
2088 }

```

```

2089 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
2090 {
2091   \msg_warning:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2092   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2093 }
2094 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
2095 {
2096   \msg_term:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2097   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2098 }
2099 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
2100 {
2101   \msg_warning:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2102   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2103 }

```

(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:`.

```

2104 \cs_new_protected:Nn \__enumext_check_ans_log:
2105 {
2106   \int_case:nn { \g__enumext_item_answer_diff_int }
2107   {
2108     { -1 } { \__enumext_check_ans_log_msg_less: }
2109     { 0 } { \__enumext_check_ans_log_msg_same_ok: }
2110     { 1 } { \__enumext_check_ans_log_msg_greater: }
2111   }
2112 }
2113 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
2114 {
2115   \msg_log:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2116   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2117 }
2118 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
2119 {
2120   \msg_log:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2121   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2122 }
2123 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
2124 {
2125   \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2126   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2127 }

```

(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.

```

2128 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
2129 {
2130   \int_compare:nNnT
2131   { \g__enumext_check_starred_cmd_int } = { 0 }
2132   {
2133     \msg_warning:nnnV
2134     { enumext } { missing-starred } { #1 } \l__enumext_check_start_line_env_tl
2135   }
2136   \int_compare:nNnT
2137   { \g__enumext_check_starred_cmd_int } > { 1 }
2138   {
2139     \msg_warning:nnnV
2140     { enumext } { many-starred } { #1 } \l__enumext_check_start_line_env_tl
2141   }
2142   \int_gzero:N \g__enumext_check_starred_cmd_int
2143   \tl_clear:N \l__enumext_check_start_line_env_tl
2144 }

```

(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*`.

```

2145 \cs_set_protected:Npn \__enumext_tmp:n #1
2146 {
2147   \keys_define:nn { enumext / #1 }
2148   {
2149     wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
2150     wrap-ans .initial:n =
2151       {
2152         \fbox{\parbox[t]{\dimeval{\itemwidth -2\fboxsep -2\fboxrule}}{##1}}
2153       },
2154     wrap-ans .value_required:n = true,
2155     wrap-opt .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
2156     wrap-opt .initial:n = [{##1}],
2157     wrap-opt .value_required:n = true,
2158     save-sep .tl_set:N = \__enumext_store_keyans_item_opt_sep_tl,
2159     save-sep .initial:n = {, ~ },
2160     save-sep .value_required:n = true,
2161     mark-ans .tl_set:N = \__enumext_mark_answer_sym_tl,
2162     mark-ans .initial:n = \textasteriskcentered,
2163     mark-ans .value_required:n = true,
2164     mark-pos .choice:,
2165     mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
2166     mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
2167     mark-pos / unknown .code:n =
2168       \msg_error:nneee { enumext } { unknown-choice }
2169       { mark-pos } { left, ~ right } { \exp_not:n {##1} },
2170     mark-pos .initial:n = right,
2171     mark-pos .value_required:n = true,
2172     show-ans .bool_set:N = \__enumext_show_answer_bool,
2173     show-ans .initial:n = false,
2174     show-ans .value_required:n = true,
2175     show-pos .bool_set:N = \__enumext_show_position_bool,
2176     show-pos .initial:n = false,
2177     show-pos .value_required:n = true,
2178     mark-ref .tl_set:N = \__enumext_mark_ref_sym_tl,
2179     mark-ref .initial:n = \textasteriskcentered,
2180     mark-ref .value_required:n = true,
2181     save-ref .bool_set:N = \__enumext_store_ref_key_bool,
2182     save-ref .initial:n = false,
2183     save-ref .value_required:n = true,
2184   }
2185 }
2186 \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`.

```

2187 \cs_set_protected:Npn \__enumext_tmp:n #1
2188 {
2189   \keys_define:nn { enumext / #1 }
2190   {
2191     mark-pos .choice:,
2192     mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
2193     mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
2194     mark-pos .initial:n = right,
2195     mark-pos .value_required:n = true,
2196     show-ans .bool_set:N = \__enumext_show_answer_bool,
2197     show-ans .initial:n = false,
2198     show-ans .value_required:n = true,
2199     show-pos .bool_set:N = \__enumext_show_position_bool,
2200     show-pos .initial:n = false,
2201     show-pos .value_required:n = true,
2202   }
2203 }
2204 \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)*.

```

2205 \cs_new_protected:Npn __enumext_store_active_keys:n #1
2206 {
2207   \bool_if:cF { __enumext_store_save_key_ __enumext_level: _bool }
2208   {
2209     \tl_clear:c { __enumext_save_key_ __enumext_level: _tl }
2210     \tl_set:ce
2211       { __enumext_store_save_key_ __enumext_level: _tl }
2212       { __enumext_filter_save_key:n {#1} }
2213   }
2214 }
2215 \cs_new_protected:Npn __enumext_store_active_keys_vii:n #1
2216 {
2217   \bool_if:NF \__enumext_store_save_key_vii_bool
2218   {
2219     \tl_clear:N \__enumext_store_save_key_vii_tl
2220     \tl_set:Ne \__enumext_store_save_key_vii_tl { __enumext_filter_save_key:n {#1} }
2221   }
2222 }

```

(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.

```

2223 \cs_set_protected:Npn __enumext_tmp:n #1
2224 {
2225   \keys_define:nn { enumext / enumext* }
2226   {
2227     save-key .code:n = __enumext_parse_save_key_vii:n {##1},
2228     save-key .value_required:n = true,
2229   }
2230   \keys_define:nn { enumext / #1 }
2231   {
2232     save-key .code:n = __enumext_parse_save_key:n {##1},
2233     save-key .value_required:n = true,
2234   }
2235 }
2236 \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*`.

```

2237 \cs_new_protected:Npn __enumext_parse_save_key:n #1
2238 {
2239   \bool_set_true:c { __enumext_store_save_key_ __enumext_level: _bool }
2240   \tl_clear:c { __enumext_save_key_ __enumext_level: _tl }
2241   \tl_set:ce
2242     { __enumext_store_save_key_ __enumext_level: _tl }
2243     { __enumext_filter_save_key:n {#1} }
2244 }

```

```

2245 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2246 {
2247   \bool_set_true:N \l__enumext_store_save_key_vii_bool
2248   \tl_clear:N \l__enumext_store_save_key_vii_tl
2249   \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2250 }

```

(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

```

```

2251 \cs_new:Npn \__enumext_filter_save_key:n #1
2252 {
2253   \use:e
2254   {
2255     \keyval_parse:NNn
2256     \__enumext_filter_save_key_key:n
2257     \__enumext_filter_save_key_pair:nn {#1}
2258   }
2259 }

```

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.

```

2260 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2261 {
2262   \str_case:nnF {#1}
2263   {
2264     { resume } {} { resume* } {} { no-store } {} { base-fix } {}
2265   }
2266   { , { \exp_not:n {#1} } }
2267 }

```

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.

```

2268 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2269 {
2270   \str_case:nnF {#1}
2271   {
2272     { series } {} { resume } {} { save-ans } {} { save-ref } {}
2273     { save-key } {} { check-ans } {} { show-ans } {} { show-pos } {}
2274     { wrap-ans } {} { mark-ans } {} { wrap-opt } {} { save-sep } {}
2275     { mark-ref } {} { mini-env } {} { mini-sep } {} { mini-right } {}
2276     { mini-right* } {}
2277   }
2278   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2279 }

```

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

12.26.4 Function for storing content in prop list

```

\__enumext_store_addto_prop:n
\__enumext_store_addto_prop:V

```

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

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

```

2280 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2281 {
2282   \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2283   {
2284     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2285   }
2286   { #1 }
2287 }
2288 \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.

```
2289 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2290 {
2291   \seq_gput_right:cn { g__enumext_ \__enumext_store_name_tl_seq } { #1 }
2292 }
2293 \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.

```
2294 \cs_new_protected:Nn \__enumext_store_level_open:
2295 {
2296   \bool_if:NT \__enumext_check_answers_bool
2297   {
2298     \tl_if_empty:CTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2299     {
2300       \__enumext_store_addto_seq:n
2301       {
2302         \item \begin{enumext}
2303       }
2304     }
2305     {
2306       \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2307       {
2308         \item \begin{enumext} [
2309       }
2310       \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2311       {
2312         ]
2313       }
2314       \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2315     }
2316   }
2317 }
2318 \cs_new_protected:Nn \__enumext_store_level_close:
2319 {
2320   \bool_if:NT \__enumext_check_answers_bool
2321   {
2322     \__enumext_store_addto_seq:n { \end{enumext} }
2323   }
2324 }
```

(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.

```
2325 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2326 {
2327   \bool_if:NT \__enumext_check_answers_bool
2328   {
2329     \tl_if_empty:NTF l__enumext_store_save_key_vii_tl
2330     {
2331       \__enumext_store_addto_seq:n
2332       {
2333         \item \begin{enumext*}
2334       }
2335     }
2336     {
2337       \tl_put_left:Nn l__enumext_store_save_key_vii_tl
2338       {
2339         \item \begin{enumext*}[
2340       }
2341       \tl_put_right:Nn l__enumext_store_save_key_vii_tl
```



```

2342         {
2343         }
2344     }
2345     \__enumext_store_addto_seq:V \__enumext_store_save_key_vii_tl
2346 }
2347 }
2348 }
2349 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2350 {
2351     \bool_if:NT \__enumext_check_answers_bool
2352     {
2353         \__enumext_store_addto_seq:n { \end{enumext*} }
2354     }
2355 }

```

(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

```

2356 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2357 {
2358     \mode_leave_vertical:
2359     \skip_horizontal:n { -\dim_use:N #2 }
2360     \makebox[0pt][ r ]
2361     {
2362         \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2363         {
2364             \tl_use:N \l__enumext_mark_answer_sym_tl
2365         }
2366     }
2367     \skip_horizontal:n { \dim_use:N #2 }
2368 }
2369 \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.

```

2370 \cs_new_protected:Nn \__enumext_store_internal_ref:
2371 {
2372     \cs_set_protected:Npn \__enumext_tmp:n ##1
2373     {
2374         \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2375         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2376         \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2377         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2378     }
2379     \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2380     \cs_set:Npn \__enumext_tmp:n ##1
2381     { . \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.

```

2382     \bool_lazy_all:nT
2383     {
2384         { \bool_if_p:N \g__enumext_starred_bool }
2385         { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2386     }
2387     {
2388         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2389         { \tl_use:N \l__enumext_label_copy_vii_tl }
2390     }

```

```

2391 \bool_lazy_all:nT
2392 {
2393   { \bool_not_p:n { \g__enumext_standar_bool } }
2394   { \bool_if_p:N \l__enumext_standar_bool }
2395   { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } } }
2396 }
2397 {
2398   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2399   {
2400     \tl_use:N \l__enumext_label_copy_vii_tl
2401     \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2402   }
2403 }

```

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.

```

2404 \bool_lazy_all:nT
2405 {
2406   { \bool_if_p:N \g__enumext_standar_bool }
2407   { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } } }
2408 { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } } }
2409 }
2410 {
2411   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2412   {
2413     \tl_use:N \l__enumext_label_copy_i_tl
2414     \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2415   }
2416 }
2417 \cs_set:Npn \__enumext_tmp:n ##1
2418 { \tl_use:c { \l__enumext_label_copy_ \int_to_roman:n {##1} _tl } . }
2419 \bool_lazy_all:nT
2420 {
2421   { \bool_if_p:N \g__enumext_standar_bool }
2422   { \bool_if_p:N \l__enumext_starred_bool }
2423   { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } } }
2424 }
2425 {
2426   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2427   {
2428     \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2429     \tl_use:N \l__enumext_label_copy_vii_tl
2430   }
2431 }

```

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

```

2432 \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2433 {
2434   \l__enumext_store_name_tl \c_colon_str
2435   \int_eval:n { \prop_count:c { \g__enumext_ \l__enumext_store_name_tl _prop } }
2436 }

```

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.

```

2437 \tl_put_right:Ne \l__enumext_write_aux_file_tl
2438 {
2439   \__enumext_newlabel:nn
2440   { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2441   { \l__enumext_newlabel_arg_two_tl }
2442 }
2443 \l__enumext_write_aux_file_tl
2444 }

```

(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*” $\langle \textit{argument} \rangle$.

```

2445 \cs_new_protected:Npn \__enumext_store_anskey_code:n #1
2446 {

```

```

2447 \int_gincr:N \g__enumext_item_anskey_int
2448 \__enumext_store_addto_prop:n {#1}
2449 \bool_if:NT \l__enumext_store_ref_key_bool
2450 {
2451   \__enumext_store_internal_ref:
2452 }
2453 \__enumext_anskey_show_wrap_left:n { #1 }

```

Now we start processing the [$\langle key = val \rangle$] passed to the command to build our `\item` in the variable `\l__enumext_store_anskey_arg_tl` which we will “store” in the $\langle sequence \rangle$. First we clear the variable `\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 `\columnbreak` and then `\item`.

```

2454 \tl_clear:N \l__enumext_store_anskey_arg_tl
2455 \bool_lazy_and:nnT
2456 { \bool_if_p:N \l__enumext_store_columns_break_bool }
2457 { \bool_not_p:n { \l__enumext_starred_bool } }
2458 {
2459   \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2460 }
2461 \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 `\l__enumext_store_anskey_arg_tl`.

```

2462 \bool_lazy_and:nnT
2463 { \bool_not_p:n { \l__enumext_starred_bool } }
2464 { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2465 {
2466   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2467   {
2468     ( \exp_not:V \l__enumext_store_item_join_int )
2469   }
2470 }

```

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

```

2471 \bool_if:NTF \l__enumext_store_item_star_bool
2472 {
2473   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2474   \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2475   {
2476     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2477     {
2478       [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2479     }
2480   }
2481   \dim_compare:nT
2482   {
2483     \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2484   }
2485   {
2486     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2487     {
2488       [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2489     }
2490   }
2491   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2492 }
2493 {
2494   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2495 }

```

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

```

2496 \bool_lazy_and:nnT
2497 { \bool_if_p:N \l__enumext_store_ref_key_bool }
2498 { \bool_if_p:N \l__enumext_hyperref_bool }
2499 {
2500   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2501   {
2502     \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2503     { \exp_not:V \l__enumext_mark_ref_sym_tl }
2504   }

```

```

2505     }
2506     \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2507 }

```

(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.

```

2508 \cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
2509 {
2510   \par
2511   \bool_if:NTF \l__enumext_starred_bool
2512   {
2513     \__enumext_print_keyans_box:NN \l__enumext_labelwidth_vii_dim \l__enumext_labelsep_vii_dim
2514   }
2515   {
2516     \__enumext_print_keyans_box:cc
2517     { \l__enumext_labelwidth_ \l__enumext_level: _dim }
2518     { \l__enumext_labelsep_ \l__enumext_level: _dim }
2519   }
2520   \__enumext_anskey_wrapper:n { #1 }
2521 }

```

(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.

```

2522 \cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
2523 {
2524   \bool_if:NT \l__enumext_show_answer_bool
2525   {
2526     \__enumext_anskey_show_wrap_arg:n { #1 }
2527   }
2528   \bool_if:NT \l__enumext_show_position_bool
2529   {
2530     \tl_set:Nx \l__enumext_mark_answer_sym_tl
2531     {
2532       \group_begin:
2533       \exp_not:N \normalfont
2534       \exp_not:N \footnotesize [ \int_eval:n
2535       {
2536         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2537       }
2538       ]
2539       \group_end:
2540     }
2541     \__enumext_anskey_show_wrap_arg:n { #1 }
2542   }
2543 }

```

(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

```

2544 \keys_define:nn { enumext / anskey }
2545 {
2546   break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2547   break-col .default:n = true,
2548   break-col .value_forbidden:n = true,
2549   item-join .int_set:N = \l__enumext_store_item_join_int,
2550   item-join .value_required:n = true,
2551   item-star .bool_set:N = \l__enumext_store_item_star_bool,
2552   item-star .default:n = true,

```

```

2553     item-star .value_forbidden:n = true,
2554     item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
2555     item-sym* .value_required:n = true,
2556     item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
2557     item-pos* .value_required:n = true,
2558     unknown .code:n = { \l__enumext_anskey_unknown:n {#1} },
2559 }

```

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`.

```

2560 \cs_new_protected:Npn \l__enumext_anskey_unknown:n #1
2561 {
2562     \exp_args:NV \l__enumext_anskey_unknown:nn \l_keys_key_str {#1}
2563 }
2564 \cs_new_protected:Npn \l__enumext_anskey_unknown:nn #1 #2
2565 {
2566     \tl_if_blank:nTF {#2}
2567     {
2568         \msg_error:nnn { enumext } { anskey-cmd-key-unknown } {#1}
2569     }
2570     {
2571         \msg_error:nnnn { enumext } { anskey-cmd-key-value-unknown } {#1} {#2}
2572     }
2573 }

```

(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`.

```

2574 \NewDocumentCommand \anskey { o +m }
2575 {
2576     \l__enumext_anskey_safe_outer:
2577     \group_begin:
2578         \bool_if:NT \l__enumext_check_answers_bool
2579         {
2580             \tl_if_novalue:nF {#1}
2581             {
2582                 \keys_set:nn { enumext / anskey } {#1}
2583             }
2584             \tl_if_blank:nTF {#2}
2585             {
2586                 \msg_error:nn { enumext } { anskey-empty-arg }
2587             }
2588             {
2589                 \l__enumext_anskey_safe_inner:
2590                 \l__enumext_store_anskey_code:n {#2}
2591             }
2592         }
2593     \group_end:
2594 }

```

(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:`

```

2595 \cs_new_protected:Nn \l__enumext_anskey_safe_outer:
2596 {
2597     \bool_if:NF \l__enumext_store_active_bool
2598     {
2599         \msg_error:nnnn { enumext } { anskey-wrong-place } { anskey } { enumext }
2600     }
2601     \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
2602     {
2603         \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans }
2604     }
2605 }

```

```

2604     }
2605     \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
2606     {
2607         \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans* }
2608     }
2609     \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
2610     {
2611         \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyanspic }
2612     }
2613 }

```

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.

```

2614 \cs_new_protected:Nn \__enumext_anskey_safe_inner:
2615 {
2616     \int_incr:N \l__enumext_anskey_level_int
2617     \int_compare:nNt { \l__enumext_anskey_level_int } > { 1 }
2618     {
2619         \msg_error:nn { enumext } { anskey-nested }
2620     }
2621     \bool_if:NF \l__enumext_item_number_bool
2622     {
2623         \msg_error:nn { enumext } { anskey-unnumber-item }
2624     }
2625     \mode_if_math:T
2626     {
2627         \msg_error:nne { enumext } { anskey-math-mode } { \c_backslash_str anskey }
2628     }
2629 }

```

(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.

```

2630 \cs_new_protected:Nn \__enumext_undefine_anskey_env:
2631 {
2632     \cs_undefine:c { anskey* }
2633     \cs_undefine:c { endanskey* }
2634     \cs_undefine:c { __scontents_anskey*_env_begin: }
2635     \cs_undefine:c { __scontents_anskey*_env_end: }
2636 }

```

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

```

2637 \__enumext_before_env:nn { enumext }
2638 {
2639     \bool_lazy_and:nnT
2640     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2641     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2642     {
2643         \cs_if_free:cF { __scontents_anskey*_env_begin: }
2644         {
2645             \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2646         }
2647     }

```

```

2648 }
2649 \__enumext_before_env:nn { enumext* }
2650 {
2651   \bool_lazy_and:nnT
2652   { \int_compare_p:nNn { \__enumext_level_int } = { 0 } }
2653   { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
2654   {
2655     \cs_if_free:cF { __scontents_anskey*_env_begin: }
2656     {
2657       \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2658     }
2659   }
2660 }

```

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.

```

2661 \__enumext_before_env:nn { anskey* }
2662 {
2663   \int_compare:nNnT { \__enumext_keyans_level_int } = { 1 }
2664   {
2665     \msg_error:nnn { enumext } { anskey-env-wrong } { keyans }
2666   }
2667   \int_compare:nNnT { \__enumext_keyans_level_h_int } = { 1 }
2668   {
2669     \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
2670   }
2671   \int_compare:nNnT { \__enumext_keyans_pic_level_int } = { 1 }
2672   {
2673     \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
2674   }
2675   \bool_if:NF \__enumext_item_number_bool
2676   {
2677     \msg_error:nn { enumext } { anskey-unnumber-item }
2678   }
2679   \mode_if_math:T
2680   {
2681     \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
2682   }
2683 }

```

(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`.

```

2684 \cs_new_protected:Npn \__enumext_anskey_env_make:n #1
2685 {
2686   \bool_if:NT \__enumext_anskey_env_bool
2687   {
2688     \newenvsc{anskey*}[store-env=#1,print-env=false]
2689     \__enumext_anskey_env_exec:
2690   }
2691 }
2692 \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`.

```

2693 \cs_new_protected:Nn \__enumext_anskey_env_define_keys:
2694 {
2695   \keys_define:nn { scontents / scontents }
2696   {
2697     break-col .bool_gset:N = \g__enumext_store_columns_break_bool,
2698     break-col .default:n = true,
2699     break-col .value_forbidden:n = true,
2700     item-join .int_gset:N = \g__enumext_store_item_join_int,
2701     item-join .value_required:n = true,

```



```

2702     item-star .bool_gset:N = \g__enumext_store_item_star_bool,
2703     item-star .default:n   = true,
2704     item-star .value_forbidden:n = true,
2705     item-sym* .tl_gset:N   = \g__enumext_store_item_symbol_tl,
2706     item-sym* .value_required:n = true,
2707     item-pos* .dim_gset:N   = \g__enumext_store_item_symbol_sep_dim,
2708     item-pos* .value_required:n = true,
2709     print-env .undefine:,
2710     store-env .undefine:,
2711     write-out .undefine:,
2712     unknown   .code:n      = { \__enumext_anskey_env_unknown:n {##1} },
2713   }
2714 }

```

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`.

```

2715 \cs_new_protected:Npn \__enumext_anskey_env_unknown:n #1
2716 {
2717   \exp_args:NV \__enumext_anskey_env_unknown:nn \l_keys_key_str {#1}
2718 }
2719 \cs_new_protected:Npn \__enumext_anskey_env_unknown:nn #1#2
2720 {
2721   \tl_if_blank:nTF {#2}
2722   {
2723     \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
2724   }
2725   {
2726     \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
2727   }
2728 }

```

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`.

```

2729 \cs_new_protected:Nn \__enumext_anskey_env_reset_keys:
2730 {
2731   \keys_define:nn { scontents / scontents }
2732   {
2733     break-col .undefine:,
2734     item-join .undefine:,
2735     item-star .undefine:,
2736     item-sym* .undefine:,
2737     item-pos* .undefine:,
2738     write-out .code:n   = {
2739       \bool_set_false:N \l__scontents_storing_bool
2740       \bool_set_true:N  \l__scontents_writing_bool
2741       \tl_set:Nn \l__scontents_fname_out_tl {##1}
2742     },
2743     write-out .value_required:n = true,
2744     print-env .meta:nn         = { scontents } { print-env = ##1 },
2745     print-env .default:n       = true,
2746     store-env .meta:nn         = { scontents } { store-env = ##1 },
2747     unknown   .code:n          = { \__scontents_parse_environment_keys:n {##1} },
2748   }
2749 }

```

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*.

```

2750 \cs_new_protected:Npn \__enumext_rescan_anskey_env:n #1
2751 {
2752   \group_begin:
2753     \int_set:Nn \tex_newlinechar:D { `^^J }
2754     \__scontents_rescan_tokens:x
2755     {
2756       \endgroup % This assumes \catcode`\=0... Things might go off otherwise.
2757       #1
2758     }
2759 }

```

(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)*.

```

2760 \cs_new_protected:Nn \__enumext_anskey_env_exec:
2761 {
2762   \__enumext_before_env:nn { anskey* }
2763   {
2764     \__enumext_anskey_env_define_keys:
2765   }

```

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.

```

2766   \hook_if_empty:nF {env/anskey*/after}
2767   {
2768     \hook_gremove_code:nn {env/anskey*/after} { * }
2769   }
2770   \__enumext_after_env:nn { anskey* }
2771   {
2772     \__enumext_anskey_env_save_keys:
2773     \tl_clear:N \l__enumext_store_anskey_env_tl
2774     \tl_clear:N \l__enumext_store_anskey_opt_tl
2775     \bool_if:NT \l__enumext_check_answers_bool
2776     {
2777       \tl_gset:Ne \l__enumext_store_anskey_env_tl
2778       {
2779         \seq_item:ce { g__scontents_name_ \l__enumext_store_name_tl _seq } { -1 }
2780       }
2781       \regex_match:nVTF
2782       { ^\s* \z | ^\s* \u{c__scontents_hidden_space_str} \z }
2783       \l__enumext_store_anskey_env_tl
2784       {
2785         \msg_error:nn { enumext } { anskey-empty-arg }
2786       }
2787       {
2788         \__enumext_anskey_env_store:
2789       }
2790     }
2791     \__enumext_anskey_env_clean_vars:
2792     \__enumext_anskey_env_reset_keys:
2793   }
2794 }

```

• 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:`

```

2795 \cs_new_protected:Nn \__enumext_anskey_env_save_keys:
2796 {
2797   \bool_lazy_and:nnT
2798   { \bool_if_p:N \g__enumext_store_columns_break_bool }
2799   { \bool_not_p:n { \l__enumext_starred_bool } }
2800   {
2801     \tl_put_left:N \l__enumext_store_anskey_opt_tl { ,break-col, }
2802   }

```

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

```

2803   \bool_lazy_and:nnT
2804   { \bool_not_p:n { \l__enumext_starred_bool } }
2805   { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }
2806   {
2807     \tl_put_left::Ne \l__enumext_store_anskey_opt_tl
2808     {
2809       ,item-join = \exp_not:V \g__enumext_store_item_join_int,
2810     }
2811   }

```

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

```

2812   \bool_if:NT \g__enumext_store_item_star_bool
2813   {
2814     \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2815     {
2816       ,item-star,
2817     }
2818     \tl_if_empty:NF \g__enumext_store_item_symbol_tl
2819     {
2820       \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2821       {
2822         ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,
2823       }
2824     }
2825     \dim_compare:nT
2826     {
2827       \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
2828     }
2829     {
2830       \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2831       {
2832         ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
2833       }
2834     }
2835   }
2836 }

```

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`.

```

2837 \cs_new_protected:Nn \__enumext_anskey_env_store:
2838 {
2839   \group_begin:
2840   \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
2841   {
2842     \exp_args:Ne
2843     \__enumext_store_anskey_code:n
2844     {
2845       \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2846     }
2847   }
2848   {
2849     \keys_set_known:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
2850     \exp_args:Ne
2851     \__enumext_store_anskey_code:n
2852     {
2853       \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2854     }
2855   }
2856   \group_end:
2857 }

```

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

```

2858 \cs_new_protected:Nn \__enumext_anskey_env_clean_vars:
2859 {
2860   \bool_gset_false:N \g__enumext_store_columns_break_bool
2861   \int_gzero:N       \g__enumext_store_item_join_int
2862   \bool_gset_false:N \g__enumext_store_item_star_bool
2863   \tl_gclear:N       \g__enumext_store_item_symbol_tl
2864   \dim_gzero:N       \g__enumext_store_item_symbol_sep_dim
2865 }

```

(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.

```

2866 \cs_new_protected:Nn \__enumext_execute_after_env:
2867 {
2868   \int_compare:nNnT { \__enumext_level_int } = { 0 }
2869   {
2870     \tl_if_empty:NF \g__enumext_store_name_tl
2871     {
2872       \__enumext_stop_save_ans_msg:
2873       \__enumext_item_answer_diff:
2874       \__enumext_log_global_vars:
2875       \__enumext_log_answer_vars:
2876       \bool_if:NTF \g__enumext_check_ans_key_bool
2877       {
2878         \__enumext_check_ans_show:
2879       }
2880       { \__enumext_check_ans_log: }
2881       \__enumext_undefine_anskey_env:
2882     }
2883     \__enumext_reset_global_vars:
2884   }
2885 }

```

(End of definition for `__enumext_execute_after_env:`.)

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

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

12.32.1 Storing content in prop list

`__enumext_keyans_addto_prop:n`

The function `__enumext_keyans_addto_prop:n` will pass the contents of the current `<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`.

```

2886 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2887 {
2888   \tl_clear:N \l__enumext_store_current_label_tl
2889   \int_compare:nNnTF { \__enumext_keyans_pic_level_int } = { 1 }
2890   {
2891     \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
2892   }
2893   {
2894     \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
2895   }
2896   \tl_if_novalue:NF { #1 }
2897   {
2898     % Set save-sep
2899     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2900     {
2901       \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
2902     }
2903     \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2904   }
2905   \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
2906 }

```

(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.

```

2907 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2908 {
2909   \bool_if:NT \l__enumext_store_ref_key_bool
2910   {
2911     \cs_set_protected:Npn \__enumext_tmp:n #1
2912     {
2913       \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2914       \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2915       \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2916       \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2917     }
2918     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2919     \__enumext_keyans_store_ref_aux_i:
2920   }
2921 }

```

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`.

```

2922 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2923 {
2924   \bool_if:NT \g__enumext_starred_bool
2925   {
2926     \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2927   }
2928   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2929   {
2930     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2931     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
2932   }
2933   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2934   {
2935     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2936     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2937   }
2938   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2939   {
2940     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2941     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2942   }
2943   \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2944   {
2945     \l__enumext_store_name_tl \c_colon_str
2946     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2947   }
2948   \__enumext_keyans_store_ref_aux_ii:
2949 }

```

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.

```

2950 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2951 {
2952   \tl_put_right:Ne \l__enumext_write_aux_file_tl
2953   {
2954     \__enumext_newlabel:nn
2955     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2956     { \l__enumext_newlabel_arg_two_tl }
2957   }
2958   \l__enumext_write_aux_file_tl
2959 }

```

(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.

```

2960 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1

```

```

2961 {
2962   \tl_clear:N \l__enumext_store_current_label_tl
2963   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2964   {
2965     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
2966   }
2967   {
2968     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
2969   }
2970   \tl_if_novalue:nF { #1 }
2971   {
2972     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2973     {
2974       \tl_put_right:Ne \l__enumext_store_current_label_tl
2975       {
2976         \l__enumext_store_keyans_item_opt_sep_tl
2977       }
2978     }
2979     \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2980   }
2981   \l__enumext_keyans_addto_seq_link:
2982 }

```

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 `\l__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 `\l__enumext_check_starred_cmd:n` and increment the value of the integer variable `\g__enumext_item_anskey_int` handled by the `check-ans` key.

```

2983 \cs_new_protected:Nn \l__enumext_keyans_addto_seq_link:
2984 {
2985   \bool_lazy_and:nnT
2986   { \bool_if_p:N \l__enumext_store_ref_key_bool }
2987   { \bool_if_p:N \l__enumext_hyperref_bool }
2988   {
2989     \tl_put_right:Ne \l__enumext_store_current_label_tl
2990     {
2991       \hfill \exp_not:N \hyperlink
2992       {
2993         \exp_not:V \l__enumext_newlabel_arg_one_tl
2994       }
2995       { \exp_not:V \l__enumext_mark_ref_sym_tl }
2996     }
2997   }
2998   \l__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
2999   \bool_if:NT \l__enumext_check_answers_bool
3000   {
3001     \int_gincr:N \g__enumext_item_anskey_int
3002   }
3003 }

```

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

```

\l__enumext_keyans_show_left:n
\l__enumext_keyans_show_ans:
\l__enumext_keyans_show_pos:
\l__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.

```

3004 \cs_new_protected:Npn \l__enumext_keyans_show_left:n #1
3005 {
3006   \tl_if_novalue:nF { #1 }
3007   {
3008     \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
3009   }
3010   \bool_if:NT \l__enumext_show_answer_bool
3011   {
3012     \l__enumext_keyans_show_ans:
3013   }
3014   \bool_if:NT \l__enumext_show_position_bool
3015   {

```

```

3016         \__enumext_keyans_show_pos:
3017     }
3018 }
3019 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
3020 {
3021     \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
3022     {
3023         \bool_lazy_or:nnT
3024         { \bool_if_p:N \l__enumext_show_answer_bool }
3025         { \bool_if_p:N \l__enumext_show_position_bool }
3026         {
3027             \__enumext_keyans_wrapper_opt:n { \l__enumext_store_current_opt_arg_tl } \c_space_tl
3028         }
3029     }
3030 }
3031 \cs_new_protected:Nn \__enumext_keyans_show_ans:
3032 {
3033     \bool_if:NT \l__enumext_starred_bool
3034     {
3035         \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
3036         \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
3037     }
3038     \tl_put_left:Nn \l__enumext_label_v_tl
3039     {
3040         \__enumext_print_keyans_box:NN
3041         \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3042     }
3043 }
3044 \cs_new_protected:Nn \__enumext_keyans_show_pos:
3045 {
3046     \bool_if:NT \l__enumext_starred_bool
3047     {
3048         \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
3049         \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
3050     }
3051     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
3052     {
3053         \tl_set:Ne \l__enumext_mark_answer_sym_tl
3054         {
3055             \group_begin:
3056             \exp_not:N \normalfont
3057             \exp_not:N \footnotesize [ \int_eval:n
3058                 {
3059                     \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3060                 }
3061             ]
3062             \group_end:
3063         }
3064     }
3065     {
3066         \tl_set:Ne \l__enumext_mark_answer_sym_tl
3067         {
3068             \group_begin:
3069             \exp_not:N \normalfont
3070             \exp_not:N \footnotesize [ \int_eval:n
3071                 {
3072                     \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
3073                 }
3074             ]
3075             \group_end:
3076         }
3077     }
3078     \tl_put_left:Nn \l__enumext_label_v_tl
3079     {
3080         \__enumext_print_keyans_box:NN
3081         \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3082     }
3083 }

```

(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`.

```

3084 \cs_new_protected:Npn \__enumext_default_item:n #1
3085 {
3086   \tl_if_novalue:nTF {#1}
3087   {
3088     \bool_if:NT \l__enumext_check_answers_bool
3089     {
3090       \int_gincr:N \g__enumext_item_number_int
3091       \bool_set_true:N \l__enumext_item_number_bool
3092     }
3093     \bool_set_true:c { \l__enumext_wrap_label_ \__enumext_level: _bool }
3094     \__enumext_item_std:w \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
3095   }
3096   {
3097     \bool_set_eq:cc
3098     { \l__enumext_wrap_label_ \__enumext_level: _bool }
3099     { \l__enumext_wrap_label_opt_ \__enumext_level: _bool }
3100     \__enumext_item_std:w [#1] \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
3101   }
3102 }

```

(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`.

```

3103 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
3104 {
3105   \tl_if_novalue:nTF {#1}
3106   {
3107     \tl_gset_eq:Nc
3108     \g__enumext_item_symbol_aux_tl { \l__enumext_item_symbol_ \__enumext_level: _tl }
3109   }
3110   {
3111     \tl_gset:Nn \g__enumext_item_symbol_aux_tl {#1}
3112   }
3113   \tl_if_novalue:nTF {#2}
3114   {
3115     \dim_set_eq:cc
3116     { \l__enumext_item_symbol_sep_ \__enumext_level: _dim }
3117     { \l__enumext_labelsep_ \__enumext_level: _dim }
3118   }
3119   {
3120     \dim_set:cn { \l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
3121   }
3122   \bool_if:NT \l__enumext_check_answers_bool
3123   {
3124     \int_gincr:N \g__enumext_item_number_int
3125     \bool_set_true:N \l__enumext_item_number_bool

```

```

3126     }
3127     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
3128     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3129 }

```

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

```

3130 \cs_new_protected:Nn \__enumext_item_star_exec:
3131 {
3132   \tl_if_empty:cF { l__enumext_item_symbol_ \__enumext_level: _tl }
3133   {
3134     \mode_leave_vertical:
3135     \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3136     \makebox[ \opt ][ r ]{ \g__enumext_item_symbol_aux_tl }
3137     \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3138   }
3139 }

```

(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*`.

```

3140 \cs_new_protected:Nn \__enumext_redefine_item:
3141 {
3142   \RenewDocumentCommand \item { s o o }
3143   {
3144     \bool_if:nTF {##1}
3145     {
3146       \__enumext_starred_item:nn {##2} {##3}
3147     }
3148     { \__enumext_default_item:n {##2} }
3149   }
3150 }

```

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

```

3151 \cs_new_protected:Nn \__enumext_make_label:
3152 {
3153   \RenewDocumentCommand \make_label { m }
3154   {
3155     \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
3156     \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3157     \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3158     {
3159       \__enumext_item_star_exec:
3160       \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3161     }
3162     { ##1 }
3163     \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
3164     \tl_gclear:N \g__enumext_item_symbol_aux_tl
3165   }
3166 }

```

(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*
3167 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
3168 {
3169   \keys_define:nn { enumext / #1 }
3170   {
3171     item-sym* .tl_set:c = { l__enumext_item_symbol_#2_tl },
3172     item-sym* .value_required:n = true,
3173     item-sym* .initial:n = { $\star$ },
3174     item-pos* .dim_set:c = { l__enumext_item_symbol_sep_#2_dim },
3175     item-pos* .value_required:n = true,
3176   }

```

```

3177     }
3178     \clist_map_inline:nn
3179     {
3180         {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
3181     }
3182     { \__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
3183 \cs_set_protected:Npn \__enumext_tmp:n #1
3184 {
3185     \keys_define:nn { enumext / #1 }
3186     {
3187         unknown .code:n = { \__enumext_keyans_unknown_keys:n {#1} }
3188     }
3189 }
3190 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }

```

Internal functions for handling `unknown` key.

```

3191 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:n #1
3192 {
3193     \exp_args:NV \__enumext_keyans_unknown_keys:nn \l_keys_key_str {#1}
3194 }
3195 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:nn #1#2
3196 {
3197     \tl_if_blank:nTF {#2}
3198     {
3199         \msg_error:nnn { enumext } { keyans-unknown-key } {#1}
3200     }
3201     {
3202         \msg_error:nnnn { enumext } { keyans-unknown-key-value } {#1} {#2}
3203     }
3204 }

```

(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
3205 \keys_define:nn { enumext / enumext* }
3206 {
3207     unknown .code:n = { \__enumext_starred_unknown_keys:n {#1} }
3208 }

```

Internal functions for handling `unknown` key.

```

3209 \cs_new_protected:Npn \__enumext_starred_unknown_keys:n #1
3210 {
3211     \exp_args:NV \__enumext_starred_unknown_keys:nn \l_keys_key_str {#1}
3212 }
3213 \cs_new_protected:Npn \__enumext_starred_unknown_keys:nn #1#2
3214 {
3215     \tl_if_blank:nTF {#2}
3216     {
3217         \msg_error:nnn { enumext } { starred-unknown-key } {#1}
3218     }
3219     {
3220         \msg_error:nnnn { enumext } { starred-unknown-key-value } {#1} {#2}
3221     }
3222 }

```

(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.

```

3223 \cs_set_protected:Npn \__enumext_tmp:n #1
3224 {
3225   \keys_define:nn { enumext / #1 }
3226   {
3227     unknown .code:n = { \__enumext_standar_unknown_keys:n {##1} }
3228   }
3229 }
3230 \clist_map_inline:nn { level-1,level-2,level-3,level-4 } { \__enumext_tmp:n {#1} }
```

Internal functions for handling `unknown` key.

```

3231 \cs_new_protected:Npn \__enumext_standar_unknown_keys:n #1
3232 {
3233   \exp_args:NV \__enumext_standar_unknown_keys:nn \l_keys_key_str {#1}
3234 }
3235 \cs_new_protected:Npn \__enumext_standar_unknown_keys:nn #1#2
3236 {
3237   \tl_if_blank:nTF {#2}
3238   {
3239     \msg_error:nnn { enumext } { standar-unknown-key } {#1}
3240   }
3241   {
3242     \msg_error:nnnn { enumext } { standar-unknown-key-value } {#1} {#2}
3243   }
3244 }
```

(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`.

```

3245 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
3246 {
3247   \tl_if_novalue:nTF { #1 }
3248   {
3249     \bool_set_true:N \__enumext_wrap_label_v_bool
3250     \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl
3251   }
3252   {
3253     \bool_set_eq:NN \__enumext_wrap_label_v_bool \__enumext_wrap_label_opt_v_bool
3254     \__enumext_item_std:w [#1] \tl_use:N \__enumext_fake_item_indent_v_tl
3255   }
3256 }
```

(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⟩`.

```

3257 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
3258 {
3259   \tl_set_eq:NN \__enumext_store_current_label_tmp_tl \__enumext_label_v_tl
3260   \__enumext_keyans_show_left:n { #1 }
3261   \bool_set_true:N \__enumext_wrap_label_v_bool
3262   \__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⟩`.

```

3263   \tl_set_eq:NN \__enumext_label_v_tl \__enumext_store_current_label_tmp_tl
3264   \__enumext_keyans_addto_prop:n { #1 }
3265   \__enumext_keyans_store_ref:
3266   \__enumext_keyans_addto_seq:n { #1 }
3267   \int_gincr:N \g__enumext_check_starred_cmd_int
3268 }
```

(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.

```

3269 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
3270 {
3271   \RenewDocumentCommand \item { s o }
3272   {
3273     \bool_if:nTF {##1}
3274     {
3275       \peek_remove_spaces:n
3276       {
3277         \__enumext_keyans_starred_item:n {##2}
3278       }
3279     }
3280     {
3281       \__enumext_keyans_default_item:n {##2}
3282     }
3283   }
3284 }

```

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

```

3285 \cs_new_protected:Nn \__enumext_keyans_make_label:
3286 {
3287   \RenewDocumentCommand \makeLabel { m }
3288   {
3289     \tl_use:N \l__enumext_label_fill_left_v_tl
3290     \tl_use:N \l__enumext_label_font_style_v_tl
3291     \bool_if:nTF \l__enumext_wrap_label_v_bool
3292     {
3293       \__enumext_wrapper_label_v:n { ##1 }
3294     }
3295     { ##1 }
3296     \tl_use:N \l__enumext_label_fill_right_v_tl
3297   }
3298 }

```

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

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

12.37 Second argument of the lists

At this point of the code we have already programmed most the necessary tools to create a custom `list` environment, remember that the function `__enumext_start_list:nn` takes two arguments, the first one we have ready, the second one we will define for all the levels of the environment `enumext` and the environment `keyans`.

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

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

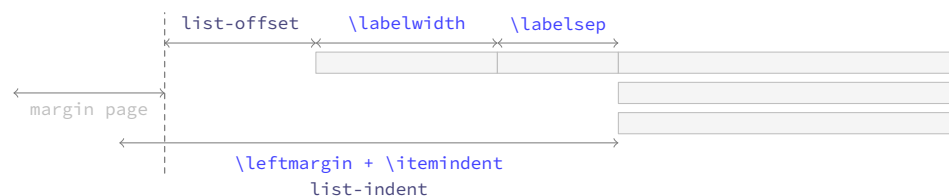


Figure 9: Representation of standard horizontal lengths in `list` environment.

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

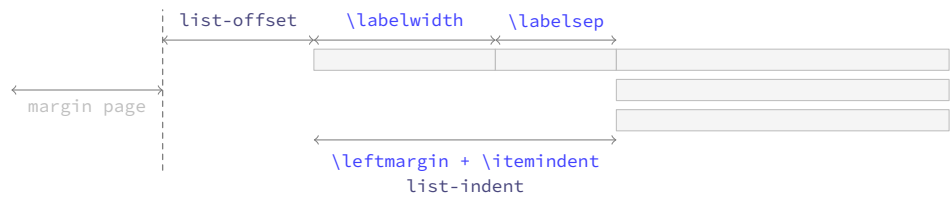
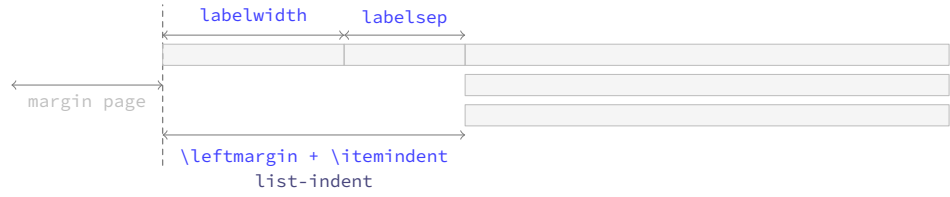
Where the default values will look like in the figure 11.

```

\__enumext_calc_hspace:NNNNNNN
\__enumext_calc_hspace:ccccccc

```

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

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

```

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

```

And returns the “adjusted” values of `\leftmargin` and `\itemindent`.

This function is passed to `__enumext_list_arg_two_X`: which is used in the definition of the `enumext` and `keyans` environments (§12.37.2).

```

3299 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNN #1 #2 #3 #4 #5 #6 #7
3300 {
3301   \dim_compare:nNnT { #1 } < { \c_zero_dim }
3302   {
3303     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
3304     \dim_set:Nn #1 { \dim_abs:n { #1 } }
3305   }
3306   \dim_compare:nNnT { #2 } < { \c_zero_dim }
3307   {
3308     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
3309     \dim_set:Nn #2 { \dim_abs:n { #2 } }
3310   }

```

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

```

3311   \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }

```

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

```

3312   \dim_compare:nNnTF { #4 } < { \c_zero_dim }
3313   {
3314     \dim_set:Nn #6 { #1 + #2 - #4 }
3315     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3316   }
3317   {
3318     \dim_compare:nNnT { #4 } = { #1 + #2 }
3319     { \dim_set:Nn #6 { \c_zero_dim } }
3320     \dim_compare:nNnT { #4 } < { #1 + #2 }
3321     { \dim_set:Nn #6 { #1 + #2 - #4 } }
3322     \dim_compare:nNnT { #4 } > { #1 + #2 }
3323     {
3324       \dim_set:Nn #6 { -#1 - #2 + #4 }
3325       \dim_set:Nn #6 { #6*-1 }
3326     }
3327     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3328   }
3329 }
3330 \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.

```

3331 \cs_set_protected:Npn \__enumext_tmp:n #1
3332 {
3333   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3334   {
3335     \__enumext_calc_hspace:ccccc
3336     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3337     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3338     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3339     { \__enumext_leftmargin_tmp_#1_bool }
3340     \clist_map_inline:nn
3341       { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3342       { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
3343     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3344       { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
3345     \usecounter { enumX#1 }
3346     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
3347     \str_if_eq:nnTF {#1} { v }
3348     {
3349       \__enumext_keyans_redefine_item:
3350       \__enumext_keyans_make_label:
3351       \__enumext_keyans_ref:
3352       \__enumext_keyans_fake_item:
3353       \bool_if:cT { \__enumext_show_length_#1_bool }
3354       {
3355         \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3356       }
3357     }
3358     {
3359       \__enumext_redefine_item:
3360       \__enumext_make_label:
3361       \__enumext_standar_ref:
3362       \__enumext_fake_item:
3363       \bool_if:cT { \__enumext_show_length_#1_bool }
3364       {
3365         \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \__enumext_level_int }
3366       }
3367     }
3368   }
3369 }
3370 \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.

```

3371 \cs_set_protected:Npn \__enumext_tmp:n #1
3372 {
3373   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3374   {
3375     \bool_set_true:c { \__enumext_leftmargin_tmp_#1_bool }
3376     \dim_zero:c { \__enumext_leftmargin_tmp_#1_dim }
3377     \__enumext_calc_hspace:ccccc
3378     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3379     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3380     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3381     { \__enumext_leftmargin_tmp_#1_bool }
3382     \clist_map_inline:nn
3383       { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3384       { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
3385     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3386       { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
3387     \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
3388     \skip_zero:N \partopsep
3389     \usecounter { enumX#1 }
3390     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }

```



```

3391     \__enumext_starred_ref:
3392     \str_if_eq:nnTF {#1} { vii }
3393     {
3394         \__enumext_fake_item_vii:
3395         \bool_if:cT { \__enumext_show_length_vii_bool }
3396         { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3397     }
3398     {
3399         \__enumext_fake_item_viii:
3400         \bool_if:cT { \__enumext_show_length_#1_bool }
3401         { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3402     }
3403 }
3404 }
3405 \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.

```

3406 \NewDocumentEnvironment{enumext}{ 0{ } }
3407 {
3408     \__enumext_safe_exec:
3409     \__enumext_parse_keys:n {#1}
3410     \__enumext_before_list:
3411     \__enumext_start_store_level:
3412     \__enumext_start_list:nn
3413     { \tl_use:c { \__enumext_label_ \__enumext_level: _tl } }
3414     {
3415         \use:c { __enumext_list_arg_two_ \__enumext_level: : }
3416         \__enumext_before_keys_exec:
3417     }
3418     \__enumext_set_item_width:
3419     \__enumext_after_args_exec:
3420 }
3421 {
3422     \__enumext_stop_list:
3423     \__enumext_stop_store_level:
3424     \__enumext_after_list:
3425 }

```

(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.

```

3426 \cs_new_protected:Nn \__enumext_set_item_width:
3427 {
3428     \dim_set:Nn \itemwidth
3429     {
3430         \linewidth
3431     }
3432     \dim_compare:nT
3433     {
3434         \dim_use:c { \__enumext_listoffset_ \__enumext_level: _dim } != \c_zero_dim
3435     }
3436     {
3437         \dim_sub:Nn \itemwidth
3438         {
3439             \dim_use:c { \__enumext_listoffset_ \__enumext_level: _dim }
3440         }
3441     }
3442 }

```

(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”.

```

3443 \cs_new_protected:Nn \__enumext_safe_exec:
3444 {
3445   \__enumext_internal_mini_page:
3446   \__enumext_is_not_nested:
3447   \int_incr:N \__enumext_level_int
3448   \int_compare:nNnT { \__enumext_level_int } > { 4 }
3449     { \msg_fatal:nn { enumext } { list-too-deep } }
3450   \bool_set_true:N \__enumext_standar_bool
3451   \bool_set_false:N \__enumext_starred_bool
3452   \__enumext_is_on_first_level:
3453 }

```

(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.

```

3454 \cs_new_protected:Npn \__enumext_parse_keys:n #1
3455 {
3456   \tl_if_novalue:nF {#1}
3457   {
3458     \str_clear:N \l__enumext_series_str
3459     \int_compare:nNnTF { \__enumext_level_int } = { 1 }
3460     {
3461       \keys_set:nn { enumext / level-1 } {#1}
3462       \__enumext_parse_series:n {#1}
3463       \__enumext_nested_base_line_fix:
3464     }
3465     {
3466       \exp_args:Ne \keys_set:nn
3467         { enumext / level-\int_use:N \__enumext_level_int } {#1}
3468     }
3469     \__enumext_store_active_keys:n {#1}
3470   }
3471 }

```

(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*`.

```

3472 \cs_new_protected:Nn \__enumext_start_store_level:
3473 {
3474   \bool_lazy_all:nT
3475   {
3476     { \bool_if_p:N \__enumext_store_active_bool }
3477     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3478     { \bool_if_p:N \g__enumext_standar_bool }
3479   }
3480   {
3481     \int_compare:nNnT { \__enumext_level_int } > { 1 }
3482     {
3483       \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3484       \__enumext_store_level_open:
3485     }
3486   }

```

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

```

3487   \bool_lazy_all:nT
3488   {
3489     { \bool_if_p:N \__enumext_store_active_bool }
3490     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3491     { \int_compare_p:nNn { \__enumext_level_h_int } = { 1 } }
3492   }
3493   {
3494     \int_compare:nNnT { \__enumext_level_int } > { 0 }
3495     {

```

```

3496         \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3497         \__enumext_store_level_open:
3498     }
3499 }
3500 }

```

Close the stored structure.

```

3501 \cs_new_protected:Nn \__enumext_stop_store_level:
3502 {
3503     \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
3504     {
3505         \__enumext_store_level_close:
3506     }
3507 }

```

(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.

```

3508 \cs_new_protected:Nn \__enumext_before_list:
3509 {
3510     \__enumext_vspace_above:
3511     \__enumext_before_args_exec:
3512     \__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.

```

3513     \dim_compare:nNt
3514     { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3515     {
3516         \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
3517         {
3518             \linewidth
3519             - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
3520             - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
3521         }

```

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.

```

3522         \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
3523         \int_gincr:N \g__enumext_minipage_stat_int
3524         \__enumext_minipage_add_space:
3525         \__enumext_mini_page{ \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
3526     }
3527     \__enumext_multicols_start:
3528 }

```

(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.

```

3529 \cs_new_protected:Nn \__enumext_multicols_start:
3530 {
3531     \int_compare:nNt
3532     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3533     {
3534         \dim_compare:nNt
3535         { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3536         {
3537             \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
3538             {
3539                 ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }

```

```

3540         + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
3541       ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
3542     - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3543   }
3544 }
3545 \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
3546 \int_compare:nNt { \l__enumext_level_int } > { 1 }
3547 {
3548   \dim_zero:N \columnseprule
3549 }

```

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.

```

3550   \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3551   {
3552     \skip_zero:N \multicolsep
3553     \__enumext_multi_addvspace:
3554   }
3555   \raggedcolumns
3556   \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
3557 }
3558 }

```

(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.

```

3559 \cs_new_protected:Nn \__enumext_multicols_stop:
3560 {
3561   \int_compare:nNt
3562   { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3563   {
3564     \end{multicols}
3565     \__enumext_unskip_unkern:
3566     \__enumext_unskip_unkern:
3567     \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
3568   }
3569 }

```

(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.

```

3570 \cs_new_protected:Nn \__enumext_after_list:
3571 {
3572   \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
3573   {
3574     \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
3575     {
3576       \msg_warning:nn { enumext } { missing-miniright }
3577       \miniright
3578     }
3579     \int_gzero:N \g__enumext_minipage_stat_int
3580     \__enumext_unskip_unkern: % remove topsep + [partopsep]
3581     \end__enumext_mini_page
3582   }
3583   {
3584     \__enumext_multicols_stop:
3585   }

```

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.

```

3586   \__enumext_after_stop_list:
3587   \__enumext_check_ans_key_hook:
3588   \__enumext_vspace_below:
3589   \bool_set_false:N \l__enumext_standar_bool

```

```

3590     \__enumext_resume_save_counter:
3591 }

```

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`.

```

3592 \__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.

```

3593 \NewDocumentEnvironment{keyans}{ 0{} }
3594 {
3595     \__enumext_keyans_safe_exec:
3596     \__enumext_keyans_parse_keys:n {#1}
3597     \__enumext_before_list_v:
3598     \__enumext_start_list:nn
3599     { \tl_use:N \l__enumext_label_v_tl }
3600     {
3601         \__enumext_list_arg_two_v:
3602         \__enumext_before_keys_exec_v:
3603     }
3604     \__enumext_keyans_set_item_width:
3605     \__enumext_after_args_exec_v:
3606 }
3607 {
3608     \__enumext_check_starred_cmd:n { item }
3609     \__enumext_stop_list:
3610     \__enumext_after_list_v:
3611 }

```

(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.

```

3612 \cs_new_protected:Nn \__enumext_keyans_set_item_width:
3613 {
3614     \dim_set:Nn \itemwidth
3615     {
3616         \linewidth
3617     }
3618     \dim_compare:nT
3619     {
3620         \l__enumext_listoffset_v_dim != \c_zero_dim
3621     }
3622     {
3623         \dim_sub:Nn \itemwidth
3624         {
3625             \l__enumext_listoffset_v_dim
3626         }
3627     }
3628 }

```

(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.

```

3629 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3630 {
3631     \bool_if:NF \l__enumext_store_active_bool
3632     {
3633         \msg_error:nnnn { enumext } { wrong-place } { keyans } { save-ans }
3634     }
3635     \int_incr:N \l__enumext_keyans_level_int
3636     \bool_set_true:N \l__enumext_keyans_env_bool
3637     \__enumext_keyans_name_and_start:

```

```

3638 % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3639 \bool_set_false:N \l__enumext_store_active_bool
3640 \int_compare:nNtT { \l__enumext_keyans_level_int } > { 1 }
3641 {
3642   \msg_error:nn { enumext } { keyans-nested }
3643 }
3644 \int_compare:nNtT { \l__enumext_level_int } > { 1 }
3645 {
3646   \msg_error:nn { enumext } { keyans-wrong-level }
3647 }
3648 }

```

(End of definition for \l__enumext_keyans_safe_exec:.)

```

\l__enumext_keyans_parse_keys:n Parse [key = val] for keyans environment.
3649 \cs_new_protected:Npn \l__enumext_keyans_parse_keys:n #1
3650 {
3651   \keys_set:nn { enumext / keyans } {#1}
3652 }

```

(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:
3653 \cs_new_protected:Nn \l__enumext_before_list_v:
3654 {
3655   \l__enumext_vspace_above_v:
3656   \l__enumext_before_args_exec_v:
3657   \dim_compare:nNtT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3658   {
3659     \dim_set:Nn \l__enumext_minipage_left_v_dim
3660     {
3661       \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3662     }
3663     \bool_set_true:N \l__enumext_minipage_active_v_bool
3664     \int_gincr:N \g__enumext_minipage_stat_int
3665     \l__enumext_keyans_minipage_add_space:
3666     \l__enumext_mini_page{ \l__enumext_minipage_left_v_dim }
3667   }
3668   \l__enumext_keyans_multicols_start:
3669 }
3670 \cs_new_protected:Nn \l__enumext_keyans_multicols_stop:
3671 {
3672   \int_compare:nNtT { \l__enumext_columns_v_int } > { 1 }
3673   {
3674     \dim_compare:nNtT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3675     {
3676       \dim_set:Nn \l__enumext_columns_sep_v_dim
3677       {
3678         (
3679           \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3680         ) / \l__enumext_columns_v_int
3681         - \l__enumext_listoffset_v_dim
3682       }
3683     }
3684     \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim
3685     \dim_zero:N \columnseprule % no rule here
3686     \bool_if:NF \l__enumext_minipage_active_v_bool
3687     {
3688       \skip_zero:N \multicolsep
3689       \l__enumext_keyans_multi_addvspace:
3690     }
3691     \raggedcolumns
3692     \begin{multicols}{\l__enumext_columns_v_int}
3693   }
3694 }
3695 \cs_new_protected:Nn \l__enumext_keyans_multicols_stop:
3696 {
3697   \int_compare:nNtT { \l__enumext_columns_v_int } > { 1 }
3698   {
3699     \end{multicols}
3700     \l__enumext_unskip_unkern:

```



```

3738     \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
3739   }

```

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

```

3740   \seq_put_right:Nn \__enumext_keyans_pic_body_seq
3741   {
3742     \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
3743   }
3744 }

```

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

`__enumext_keyans_anspic_code:nnn`

The function `__enumext_keyans_anspic_code:nnn` will be in charge of handling the “counter” and *⟨label⟩*, which will have the same configuration as the `keyans` environment.

```

3745 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
3746 {
3747   \stepcounter { enumXvi }
3748   #3 \\\
3749   \bool_if:nT { #1 }
3750   {
3751     \__enumext_keyans_addto_prop:n { #2 }
3752     \__enumext_keyans_store_ref:
3753     \__enumext_keyans_addto_seq:n { #2 }
3754     \int_gincr:N \g__enumext_check_starred_cmd_int
3755     \bool_lazy_or:nnT
3756     { \bool_if_p:N \__enumext_show_answer_bool }
3757     { \bool_if_p:N \__enumext_show_position_bool }
3758     {
3759       \tl_set_eq:NN \__enumext_label_v_tl \__enumext_label_vi_tl
3760       \__enumext_keyans_show_left:n { #2 }
3761       \tl_set_eq:NN \__enumext_label_vi_tl \__enumext_label_v_tl
3762     }
3763   }
3764   \tl_use:N \__enumext_label_font_style_v_tl
3765   \__enumext_wrapper_label_v:n { \__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
3766 }

```

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

12.40.2 The environment `keyanspic`

`keyanspic`

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

```

3767 \NewDocumentEnvironment{keyanspic}{ o }
3768 {
3769   \__enumext_keyans_pic_safe_exec:
3770   \__enumext_start_list:nn
3771   { }
3772   {
3773     \__enumext_keyans_pic_arg_two:
3774   }

```

We apply the “adjusted” vertical spacing above the environment

```

3775   \vspace { \__enumext_keyans_pic_above_skip }
3776 }

```

If the optional argument is not present, the number of times the `\anspic` command appears will be counted from `__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.

```

3777 {
3778   \tl_if_novalue:nTF { #1 }
3779   {
3780     \__enumext_keyans_pic_do:e { \seq_count:N \__enumext_keyans_pic_body_seq }
3781   }
3782   { \__enumext_keyans_pic_do:n { #1 } }
3783   \__enumext_stop_list:
3784   \__enumext_check_starred_cmd:n { anspic }
3785   \setcounter { enumXvi } { 0 }
3786   \vspace { \__enumext_topsep_v_skip }
3787   %\bool_set_false:N \__enumext_store_active_bool
3788 }

```

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

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

```

3789 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
3790 {
3791   \int_incr:N \l__enumext_keyans_pic_level_int
3792   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
3793   {
3794     \msg_error:nn { enumext } { keyanspic-nested }
3795   }
3796   \__enumext_keyans_name_and_start:
3797 }

```

(End of definition for `__enumext_keyans_pic_safe_exec:.`)

`__enumext_keyans_pic_skip_abs:N` The function `__enumext_keyans_pic_skip_abs:N` will return a positive value `\parsep`.

```

3798 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3799 {
3800   \dim_compare:nNnT { #1 } < { 0pt }
3801   { \skip_set:Nn #1 { -#1 } }
3802 }

```

(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.

```

3803 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
3804 {

```

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

```

3805   \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3806   \__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.

```

3807   \skip_add:Nn \parsep { \itemsep }
3808   \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
3809   \dim_zero:N \labelwidth
3810   \dim_zero:N \listparindent
3811   \dim_zero:N \labelsep
3812   \skip_zero:N \partopsep
3813   \skip_zero:N \itemsep

```

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

```

3814   \__enumext_keyans_pic_skip_abs:N \parsep
3815   \skip_set:Nn \l__enumext_keyans_pic_above_skip
3816   {
3817     \box_dp:N \strutbox
3818     + \l__enumext_topsep_v_skip
3819     - \parsep
3820   }
3821   \__enumext_item_std:w \scan_stop:
3822   % paranoia
3823   \RenewDocumentCommand \item {}
3824   {
3825     \msg_error:nn { enumext } { keyanspic-item-cmd }
3826   }
3827 }

```

(End of definition for `__enumext_keyans_pic_arg_two:.`)

```

\__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`.

```

3828 \cs_new_protected:Nn \__enumext_keyans_pic_do:n
3829 {
3830   \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
3831 }
3832 \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.

```

3833 \cs_new_protected:Nn \__enumext_keyans_pic_row:n
3834 {
3835   \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
3836   \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
3837   \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
3838   \int_step_inline:nnn
3839     { \l__enumext_keyans_pic_above_int + 1 }
3840     { \l__enumext_keyans_pic_below_int }
3841     {
3842       \__enumext_minipage:w [ b ] { \l__enumext_keyans_pic_width_dim }
3843       \centering
3844       \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
3845       \__enumext_endminipage:
3846     }
3847   \par
3848 }

```

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

12.41 The horizontal environments

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

12.42 Redefining `\footnote` command

```

\__enumext_footnotetext:nn
\__enumext_renew_footnote:
\__enumext_print_footnote:

```

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

```

3849 \cs_new_protected:Nn \__enumext_footnotetext:nn
3850 {
3851   \footnotetext[#1]{#2}
3852 }
3853 \cs_new_protected:Nn \__enumext_renew_footnote:
3854 {
3855   \seq_gclear:N \g__enumext_footnote_arg_seq
3856   \seq_gclear:N \g__enumext_footnote_int_seq
3857   \RenewDocumentCommand \footnote { o +m }
3858   {
3859     \tl_if_novalue:nTF {##1}
3860     {
3861       \stepcounter{footnote}
3862       \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
3863     }
3864     {
3865       \int_gset:Nn \g__enumext_footnote_int { ##1 }
3866     }
3867     \footnotemark [ \g__enumext_footnote_int ]
3868     \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
3869     \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
3870   }
3871 }
3872 \cs_new_protected:Nn \__enumext_print_footnote:
3873 {
3874   \seq_if_empty:NF \g__enumext_footnote_int_seq
3875   {
3876     \seq_map_pairwise_function:NNN
3877     \g__enumext_footnote_int_seq

```

```

3878         \g__enumext_footnote_arg_seq
3879         \__enumext_footnotetext:nn
3880     }
3881 }

```

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

12.42.1 Functions for item box width

To achieve the horizontal list environment we will capture the `\item` command and the $\langle content \rangle$ 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:
3882 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
3883 {
3884     \dim_compare:nNt { \__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3885     {
3886         \dim_set:Nn \__enumext_columns_sep_vii_dim
3887         {
3888             ( \__enumext_labelwidth_vii_dim + \__enumext_labelsep_vii_dim )
3889             / \__enumext_columns_vii_int
3890         }
3891     }
3892     \int_set:Nn \__enumext_tmpa_vii_int { \__enumext_columns_vii_int - 1 }
3893     \dim_set:Nn \__enumext_item_width_vii_dim
3894     {
3895         ( \linewidth - \__enumext_columns_sep_vii_dim * \__enumext_tmpa_vii_int )
3896         / \__enumext_columns_vii_int
3897         - \__enumext_labelwidth_vii_dim
3898         - \__enumext_labelsep_vii_dim
3899     }

```

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

```

3900     \dim_compare:nNt { \__enumext_rightmargin_vii_dim } > { \c_zero_dim }
3901     {
3902         \dim_sub:Nn \__enumext_item_width_vii_dim
3903         {
3904             ( \__enumext_rightmargin_vii_dim * \__enumext_tmpa_vii_int )
3905             / \__enumext_columns_vii_int
3906         }
3907         \dim_add:Nn \__enumext_columns_sep_vii_dim
3908         {
3909             \__enumext_rightmargin_vii_dim
3910         }
3911     }
3912 }

```

Same implementation for the `keyans*` environment.

```

3913 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3914 {
3915     \dim_compare:nNt { \__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3916     {
3917         \dim_set:Nn \__enumext_columns_sep_viii_dim
3918         {
3919             ( \__enumext_labelwidth_viii_dim + \__enumext_labelsep_viii_dim )
3920             / \__enumext_columns_viii_int
3921         }
3922     }
3923     \int_set:Nn \__enumext_tmpa_viii_int { \__enumext_columns_viii_int - 1 }
3924     \dim_set:Nn \__enumext_item_width_viii_dim
3925     {
3926         ( \linewidth - \__enumext_columns_sep_viii_dim * \__enumext_tmpa_viii_int )
3927         / \__enumext_columns_viii_int
3928         - \__enumext_labelwidth_viii_dim
3929         - \__enumext_labelsep_viii_dim
3930     }
3931     \dim_compare:nNt { \__enumext_rightmargin_viii_dim } > { \c_zero_dim }
3932     {
3933         \dim_sub:Nn \__enumext_item_width_viii_dim
3934         {

```

```

3935         ( \l__enumext_rightmargin_viii_dim * \l__enumext_tmpa_vii_int )
3936         / \l__enumext_columns_viii_int
3937     }
3938     \dim_add:Nn \l__enumext_columns_sep_viii_dim
3939     {
3940         \l__enumext_rightmargin_viii_dim
3941     }
3942 }
3943 }

```

(End of definition for \l__enumext_starred_columns_set_vii: and \l__enumext_starred_columns_set_viii:.)

12.42.2 Functions for join item columns

\l__enumext_starred_joined_item_vii:n
\l__enumext_starred_joined_item_viii:n

The functions \l__enumext_starred_joined_item_vii:n and \l__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.

```

3944 \cs_new_protected:Npn \l__enumext_starred_joined_item_vii:n #1
3945 {
3946     \int_set:Nn \l__enumext_joined_item_vii_int {#1}
3947     \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
3948     {
3949         \msg_warning:nnee { enumext } { item-joined }
3950         { \int_use:N \l__enumext_joined_item_vii_int }
3951         { \int_use:N \l__enumext_columns_vii_int }
3952         \int_set:Nn \l__enumext_joined_item_vii_int
3953         {
3954             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
3955         }
3956     }
3957     \int_compare:nNnT
3958     { \l__enumext_joined_item_vii_int }
3959     >
3960     { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
3961     {
3962         \msg_warning:nnee { enumext } { item-joined-columns }
3963         { \int_use:N \l__enumext_joined_item_vii_int }
3964         {
3965             \int_eval:n
3966             { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
3967         }
3968         \int_set:Nn \l__enumext_joined_item_vii_int
3969         {
3970             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
3971         }
3972     }
3973     \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { 1 }
3974     {
3975         \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
3976         \int_decr:N \l__enumext_joined_item_aux_vii_int
3977         \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3978         \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3979         \dim_set:Nn \l__enumext_joined_width_vii_dim
3980         {
3981             \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
3982             + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3983               + \l__enumext_columns_sep_vii_dim
3984               ) * \l__enumext_joined_item_aux_vii_int
3985         }
3986         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
3987     }
3988     {
3989         \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
3990         \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
3991     }
3992 }

```

Same implementation for the **keyans*** environment.

```

3993 \cs_new_protected:Npn \l__enumext_starred_joined_item_viii:n #1
3994 {
3995     \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3996     \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }

```

```

3997     {
3998         \msg_warning:nnee { enumext } { item-joined }
3999         { \int_use:N \l__enumext_joined_item_viii_int }
4000         { \int_use:N \l__enumext_columns_viii_int }
4001         \int_set:Nn \l__enumext_joined_item_viii_int
4002         {
4003             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4004         }
4005     }
4006     \int_compare:nNnT
4007     { \l__enumext_joined_item_viii_int }
4008     >
4009     { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4010     {
4011         \msg_warning:nnee { enumext } { item-joined-columns }
4012         { \int_use:N \l__enumext_joined_item_viii_int }
4013         {
4014             \int_eval:n
4015             { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4016         }
4017         \int_set:Nn \l__enumext_joined_item_viii_int
4018         {
4019             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4020         }
4021     }
4022     \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { 1 }
4023     {
4024         \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
4025         \int_decr:N \l__enumext_joined_item_aux_viii_int
4026         \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
4027         \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
4028         \dim_set:Nn \l__enumext_joined_width_viii_dim
4029         {
4030             \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
4031             + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
4032                 + \l__enumext_columns_sep_viii_dim
4033                 ) * \l__enumext_joined_item_aux_viii_int
4034         }
4035         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
4036     }
4037     {
4038         \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
4039         \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
4040     }
4041 }

```

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

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

```

\__enumext_start_mini_vii:
\__enumext_stop_mini_vii:

```

The implementation of the `mini-env` key support is almost identical to the one used in the `enumext` and `keyans` environments, the difference is that the `__enumext_mini_page` environment on the “right side” is executed “after” closing the environment, so it is necessary to make a global copy of the variable `\l__enumext_minipage_right_vii_dim` in the variable `\g__enumext_minipage_right_vii_dim`.

```

4042 \cs_new_protected:Nn \__enumext_start_mini_vii:
4043 {
4044     \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
4045     {
4046         \dim_set:Nn \l__enumext_minipage_left_vii_dim
4047         {
4048             \linewidth
4049             - \l__enumext_minipage_right_vii_dim
4050             - \l__enumext_minipage_hsep_vii_dim
4051         }
4052         \bool_set_true:N \l__enumext_minipage_active_vii_bool
4053         \dim_gset_eq:NN
4054         \g__enumext_minipage_right_vii_dim
4055         \l__enumext_minipage_right_vii_dim
4056         \__enumext_mini_addvspace_vii:
4057         \nointerlineskip\nindent
4058         \__enumext_mini_page{ \l__enumext_minipage_left_vii_dim }
4059     }

```

```
4060 }
```

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”.

```
4061 \cs_new_protected:Nn \__enumext_stop_mini_vii:
4062 {
4063   \bool_if:NT \l__enumext_minipage_active_vii_bool
4064   {
4065     \end__enumext_mini_page
4066     \hfill
4067     \bool_gset_true:N \g__enumext_minipage_active_vii_bool
4068   }
4069 }
```

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.

```
4070 \__enumext_after_env:nn {enumext*}
4071 {
4072   \bool_if:NT \g__enumext_minipage_active_vii_bool
4073   {
4074     \__enumext_mini_page{ \g__enumext_minipage_right_vii_dim }
4075     \par\addvspace { \g__enumext_minipage_right_skip }
4076     \bool_if:NF \g__enumext_minipage_center_vii_bool
4077     {
4078       \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
4079       {
4080         \centering
4081       }
4082     }
4083     \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
4084     {
4085       \tl_use:N \g__enumext_miniright_code_vii_tl
4086     }
4087     \box_use_drop:N \l__enumext_miniright_code_vii_box
4088     \end__enumext_mini_page
4089     \par\addvspace{ \g__enumext_minipage_after_skip }
4090   }
4091   \bool_gset_false:N \g__enumext_minipage_active_vii_bool
4092   \bool_gset_true:N \g__enumext_minipage_center_vii_bool
4093   \tl_gclear:N \g__enumext_miniright_code_vii_tl
4094   \dim_gzero:N \g__enumext_minipage_right_vii_dim
4095   \bool_gset_false:N \g__enumext_starred_bool
4096 }
```

(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.

```
4097 \cs_new_protected:Nn \__enumext_start_mini_viii:
4098 {
4099   \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
4100   {
4101     \dim_set:Nn \l__enumext_minipage_left_viii_dim
4102     {
4103       \linewidth
4104       - \l__enumext_minipage_right_viii_dim
4105       - \l__enumext_minipage_hsep_viii_dim
4106     }
4107     \bool_set_true:N \l__enumext_minipage_active_viii_bool
4108     \dim_gset_eq:NN
4109     \g__enumext_minipage_right_viii_dim
4110     \l__enumext_minipage_right_viii_dim
4111     \__enumext_mini_addvspace_viii:
4112     \nointerlineskip\noindent
4113     \__enumext_mini_page{ \l__enumext_minipage_left_viii_dim }
4114   }
4115 }
```



```

4116 \cs_new_protected:Nn \__enumext_stop_mini_viii:
4117 {
4118   \bool_if:NT \l__enumext_minipage_active_viii_bool
4119   {
4120     \end__enumext_mini_page
4121     \hfill
4122     \bool_gset_true:N \g__enumext_minipage_active_viii_bool
4123   }
4124 }
4125 \__enumext_after_env:nn {keyans*}
4126 {
4127   \bool_if:NT \g__enumext_minipage_active_viii_bool
4128   {
4129     \__enumext_mini_page{ \g__enumext_minipage_right_viii_dim }
4130     \par\addvspace { \g__enumext_minipage_right_skip }
4131     \bool_if:NF \g__enumext_minipage_center_viii_bool
4132     {
4133       \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
4134       {
4135         \centering
4136       }
4137     }
4138     \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
4139     {
4140       \tl_use:N \g__enumext_miniright_code_viii_tl
4141     }
4142     \box_use_drop:N \l__enumext_miniright_code_viii_box
4143     \end__enumext_mini_page
4144     \par\addvspace{ \g__enumext_minipage_after_skip }
4145   }
4146   \bool_gset_false:N \g__enumext_minipage_active_viii_bool
4147   \bool_gset_true:N \g__enumext_minipage_center_viii_bool
4148   \tl_gclear:N \g__enumext_miniright_code_viii_tl
4149   \dim_gzero:N \g__enumext_minipage_right_viii_dim
4150 }

```

(End of definition for __enumext_start_mini_viii: and __enumext_stop_mini_viii:.)

12.43 The environment enumext*

enumext* First we will generate the environment and we will give a temporary definition to __enumext_stop_item_tmp_vii: equal to \noindent and next to \item equal to __enumext_start_item_tmp_vii: which we will redefine later. 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.

```

4151 \NewDocumentEnvironment{enumext*}{ o }
4152 {
4153   \__enumext_safe_exec_vii:
4154   \__enumext_parse_keys_vii:n {#1}
4155   \__enumext_before_list_vii:
4156   \__enumext_start_store_level_vii:
4157   \__enumext_start_list:nn { }
4158   {
4159     \__enumext_list_arg_two_vii:
4160     \__enumext_before_keys_exec_vii:
4161   }
4162   \__enumext_starred_columns_set_vii:
4163   \item[] \scan_stop:
4164   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
4165   \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
4166   \ignorespaces
4167 }
4168 {
4169   \__enumext_stop_item_tmp_vii:
4170   \__enumext_remove_extra_parsep_vii:
4171   \__enumext_stop_list:
4172   \__enumext_stop_store_level_vii:
4173   \__enumext_after_list_vii:
4174 }

```

(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.

```

4175 \cs_new_protected:Nn \__enumext_safe_exec_vii:
4176 {
4177   \__enumext_internal_mini_page:
4178   \__enumext_is_not_nested:
4179   \int_incr:N \l__enumext_level_h_int
4180   \int_compare:nNtT { \l__enumext_level_h_int } > { 1 }
4181   {
4182     \msg_error:nn { enumext } { nested }
4183   }
4184   \int_compare:nNtT { \l__enumext_keyans_level_h_int } = { 1 }
4185   {
4186     \msg_error:nnn { enumext } { nested-horizontal } { keyans*}
4187   }
4188   \bool_set_true:N \l__enumext_starred_bool
4189   \bool_set_false:N \l__enumext_standar_bool
4190   \__enumext_is_on_first_level:
4191 }

```

(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`.

```

4192 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
4193 {
4194   \tl_if_novalue:nF {#1}
4195   {
4196     \str_clear:N \l__enumext_series_str
4197     \keys_set:nn { enumext / enumext* } {#1}
4198     \__enumext_parse_series:n {#1}
4199     \__enumext_store_active_keys_vii:n {#1}
4200     \__enumext_nested_base_line_fix:
4201   }
4202 }

```

(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*`.

```

4203 \cs_new_protected:Nn \__enumext_before_list_vii:
4204 {
4205   \__enumext_vspace_above_vii:
4206   \__enumext_check_ans_active:
4207   \__enumext_before_args_exec_vii:
4208   \__enumext_start_mini_vii:
4209 }

```

(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.

```

4210 \cs_new_protected:Nn \__enumext_after_list_vii:
4211 {
4212   \__enumext_stop_mini_vii:
4213   \__enumext_after_stop_list_vii:

```

```

4214     \__enumext_check_ans_key_hook:
4215     \__enumext_vspace_below_vii:
4216     \bool_set_false:N \__enumext_starred_bool
4217     \__enumext_resume_save_counter:
4218 }

```

(End of definition for __enumext_after_list_vii:.)

```

\__enumext_start_store_level_vii:
\__enumext_stop_store_level_vii:

```

The __enumext_start_store_level_vii: and __enumext_stop_store_level_vii: functions activate the level saving mechanism for storage in *(sequence)* of the \anskey command and anskey* environment if enumext* are nested in enumext.

```

4219 \cs_new_protected:Nn \__enumext_start_store_level_vii:
4220 {
4221     \bool_if:NT \__enumext_store_active_bool
4222     {
4223         \int_compare:nNt { \__enumext_level_int } > { 0 }
4224         {
4225             \__enumext_store_level_open_vii:
4226         }
4227     }
4228 }
4229 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
4230 {
4231     \bool_if:NT \__enumext_store_active_bool
4232     {
4233         \int_compare:nNt { \__enumext_level_int } > { 0 }
4234         {
4235             \__enumext_store_level_close_vii:
4236         }
4237     }
4238 }

```

(End of definition for __enumext_start_store_level_vii: and __enumext_stop_store_level_vii:.)

12.43.1 The command \item in enumext*

```
\__enumext_start_item_tmp_vii:
```

First we will call the function __enumext_stop_item_tmp_vii: that we will redefine later, we will increment the value of __enumext_item_column_pos_vii_int that will count the item's by rows and the value of \g__enumext_item_count_all_vii_int that will count the total of item's in the environment. After that we will call the function __enumext_item_peek_args_vii: that will handle the arguments passed to \item.

```

4239 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
4240 {
4241     \__enumext_stop_item_tmp_vii:
4242     \int_incr:N \__enumext_item_column_pos_vii_int
4243     \int_gincr:N \g__enumext_item_count_all_vii_int
4244     \__enumext_item_peek_args_vii:
4245 }

```

(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).

```

4246 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
4247 {
4248     \peek_meaning:NTF (
4249     { \__enumext_joined_item_vii:w }
4250     { \__enumext_joined_item_vii:w (1) }
4251 }

```

(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.

```

4252 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
4253 {
4254     \__enumext_starred_joined_item_vii:n {#1}
4255     \peek_meaning_remove:NTF *

```

```

4256     { \__enumext_starred_item_vii:w }
4257     { \__enumext_standar_item_vii:w }
4258 }

```

(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].

```

4259 \cs_new_protected:Npn \__enumext_standar_item_vii:w
4260 {
4261   \bool_set_false:N \l__enumext_item_starred_vii_bool
4262   \peek_meaning:NTF [
4263     {
4264       \bool_set_eq:NN
4265         \l__enumext_wrap_label_vii_bool
4266         \l__enumext_wrap_label_opt_vii_bool
4267       \__enumext_start_item_vii:w
4268     }
4269     {
4270       \bool_set_true:N \l__enumext_wrap_label_vii_bool
4271       \legacy_if_set_true:n { @noitemarg }
4272       \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4273     }
4274 }

```

(End of definition for __enumext_standar_item_vii:w.)

__enumext_starred_item_vii:w

The function __enumext_starred_item_vii:w together with the specified auxiliary functions `aux_i:w`, `aux_ii:w`, and `aux_iii:w` execute `\item*`, `\item*[⟨symbol⟩]` and `\item*[⟨symbol⟩][⟨offset⟩]`.

__enumext_starred_item_vii_aux_i:w

__enumext_starred_item_vii_aux_ii:w

__enumext_starred_item_vii_aux_iii:w

```

4275 \cs_new_protected:Npn \__enumext_starred_item_vii:w
4276 {
4277   \bool_set_true:N \l__enumext_item_starred_vii_bool
4278   \bool_set_true:N \l__enumext_wrap_label_vii_bool
4279   \peek_meaning:NTF [
4280     { \__enumext_starred_item_vii_aux_i:w }
4281     { \__enumext_starred_item_vii_aux_ii:w }
4282   }
4283   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
4284   {
4285     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
4286     \__enumext_starred_item_vii_aux_ii:w
4287   }
4288   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
4289   {
4290     \peek_meaning:NTF [
4291       { \__enumext_starred_item_vii_aux_iii:w }
4292       {
4293         \dim_set_eq:NN
4294           \l__enumext_item_symbol_sep_vii_dim
4295           \l__enumext_labelsep_vii_dim
4296         \legacy_if_set_true:n { @noitemarg }
4297         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4298       }
4299     }
4300   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
4301   {
4302     \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
4303     \legacy_if_set_true:n { @noitemarg }
4304     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4305   }

```

(End of definition for __enumext_starred_item_vii:w and others.)

12.43.2 Real definition of \item in enumext*

_enumext_start_item_vii:w

The functions _enumext_start_item_vii:w and _enumext_stop_item_vii: executing the true definition of \item inside the enumext* environment. The first thing we will do is set the value of _enumext_stop_item_tmp_vii: equal to _enumext_stop_item_vii: which we will define later and add the hyperref compatible enumXvii counter, after that we will start capturing the item content in a box. Here need setting the \ifhyper@item switch to “true” for hyperref compatible. The explanation for this is given by the master Heiko Oberdiek on \refstepcounter{enumi} twice (or more) creates destination with the same identifier.

```

4306 \cs_new_protected_nopar:Npn \_enumext_start_item_vii:w [#1]
4307 {
4308   \cs_set_eq:NN \_enumext_stop_item_tmp_vii: \_enumext_stop_item_vii:
4309   \legacy_if:nT { @noitemarg }
4310   {
4311     \legacy_if_set_false:n { @noitemarg }
4312     \legacy_if:nT { @nmbrrlist }
4313     {
4314       \bool_if:NT \l__enumext_hyperref_bool
4315       {
4316         \legacy_if_set_true:n { @hyper@item }
4317       }
4318       \refstepcounter{enumXvii}
4319       \bool_if:NT \l__enumext_check_answers_bool
4320       {
4321         \int_gincr:N \g__enumext_item_number_int
4322         \bool_set_true:N \l__enumext_item_number_bool
4323       }
4324     }
4325   }

```

Here we start capturing \item and its contents in a *horizontal box* where the width will be \itemwidth plus \labelwidth plus \labelsep, unlike the implementation in shortlst we will not use an extra group and the plain form of the lrbox environment. If the state of the variable \l__enumext_footnotes_key_bool is false, we will redefine the command \footnote, followed by printing the ⟨symbol⟩ defined for \item* if it is present and open a new group inside which we execute font key next to \item and the keys wrap-label, wrap-label*, align, close the group and execute the key labelsep and then the key first. Finally we open the minipage environment and execute the listparindent key which will be equal to \parindent, the parsep key which will be equal to \parskip and the itemindent key.

```

4326 \hbox_set_to_wd:Nnw \l__enumext_item_text_vii_box
4327 {
4328   \l__enumext_joined_width_vii_dim
4329   + \l__enumext_labelwidth_vii_dim
4330   + \l__enumext_labelsep_vii_dim
4331 }
4332 \bool_if:NF \l__enumext_footnotes_key_bool
4333 {
4334   \_enumext_renew_footnote:
4335 }
4336 \bool_if:NT \l__enumext_item_starred_vii_bool
4337 {
4338   \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
4339   {
4340     \tl_gset_eq:NN
4341     \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
4342   }
4343   \mode_leave_vertical:
4344   \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
4345   \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
4346   \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
4347   \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
4348 }
4349 \group_begin:
4350 \tl_use:N \l__enumext_label_font_style_vii_tl
4351 \bool_if:NTF \l__enumext_wrap_label_vii_bool
4352 {
4353   \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
4354   { \_enumext_wrapper_label_vii:n {#1} }
4355 }
4356 {
4357   \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }

```

```

4358     }
4359     \group_end:
4360     \skip_horizontal:N \l__enumext_labelsep_vii_dim
4361     \tl_use:N \l__enumext_after_list_args_vii_tl
4362     \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
4363     \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
4364     \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
4365     \tl_use:N \l__enumext_fake_item_indent_vii_tl
4366 }

```

(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 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.

```

4367 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
4368 {
4369     \__enumext_endminipage:
4370     \hbox_set_end:
4371     \int_set:Nn \hbadness { 10000 }
4372     \box_use_drop:N \l__enumext_item_text_vii_box
4373     \bool_if:NF \l__enumext_footnotes_key_bool
4374     {
4375         \__enumext_print_footnote:
4376     }
4377     \int_compare:nNnTF
4378     { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
4379     {
4380         \par\noindent
4381         \int_zero:N \l__enumext_item_column_pos_vii_int
4382     }
4383     {
4384         \skip_horizontal:N \l__enumext_columns_sep_vii_dim
4385     }
4386 }

```

(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*.

```

4387 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
4388 {
4389     \int_compare:nNnT
4390     {
4391         \int_mod:nn
4392         { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
4393     }
4394     =
4395     { 0 }
4396     {
4397         \para_end:
4398         \skip_vertical:n { -\l__enumext_itemsep_vii_skip }
4399         \skip_vertical:N \c_zero_skip
4400         \int_gzero:N \g__enumext_item_count_all_vii_int
4401     }
4402 }

```

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`.

```

4403 \__enumext_after_env:nn {enumext*} { \__enumext_execute_after_env: }

```

(End of definition for `__enumext_remove_extra_parsep_vii:`.)

12.44 The environment keyans*

keyans*

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

```

4404 \NewDocumentEnvironment{keyans*}{ o }
4405 {
4406   \__enumext_safe_exec_viii:
4407   \__enumext_parse_keys_viii:n {#1}
4408   \__enumext_before_list_viii:
4409   \__enumext_start_list:nn { }
4410   {
4411     \__enumext_list_arg_two_viii:
4412     \__enumext_before_keys_exec_viii:
4413   }
4414   \__enumext_starred_columns_set_viii:
4415   \item[] \scan_stop:
4416   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
4417   \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
4418   \ignorespaces
4419 }
4420 {
4421   \__enumext_stop_item_tmp_viii:
4422   \__enumext_remove_extra_parsep_viii:
4423   \__enumext_check_starred_cmd:n { item }
4424   \__enumext_stop_list:
4425   \__enumext_after_list_viii:
4426 }
```

(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 `__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.

```

4427 \cs_new_protected:Nn \__enumext_safe_exec_viii:
4428 {
4429   \bool_if:NF \__enumext_store_active_bool
4430   {
4431     \msg_error:nnnn { enumext } { wrong-place } { keyans* } { save-ans }
4432   }
4433   \int_incr:N \__enumext_keyans_level_h_int
4434   \int_compare:nNnT { \__enumext_keyans_level_h_int } > { 1 }
4435   {
4436     \msg_error:nn { enumext } { nested }
4437   }
4438   \__enumext_keyans_name_and_start:
4439   \bool_if:NT \__enumext_starred_bool
4440   {
4441     \msg_error:nnn { enumext } { nested-horizontal } { enumext* }
4442   }
4443   \bool_set_true:N \__enumext_starred_bool
4444   % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
4445   \bool_set_false:N \__enumext_store_active_bool
4446   \int_compare:nNnT { \__enumext_level_int } > { 1 }
4447   {
4448     \msg_error:nn { enumext } { keyans-wrong-level }
4449   }
4450 }
```

(End of definition for __enumext_safe_exec_viii:.)

__enumext_parse_keys_viii:n

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

```

4451 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
4452 {
4453   \tl_if_novalue:nF {#1}
```



```

4454     {
4455         \keys_set:nn { enumext / keyans* } {#1}
4456     }
4457 }

```

(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`.

```

4458 \cs_new_protected:Nn \__enumext_before_list_viii:
4459 {
4460     \__enumext_vspace_above_viii:
4461     \__enumext_before_args_exec_viii:
4462     \__enumext_start_mini_viii:
4463 }

```

(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.

```

4464 \cs_new_protected:Nn \__enumext_after_list_viii:
4465 {
4466     \__enumext_stop_mini_viii:
4467     \__enumext_after_stop_list_viii:
4468     \__enumext_vspace_below_viii:
4469 }

```

(End of definition for `__enumext_after_list_viii:`.)

12.44.1 The command `\item` in `keyans*`

The idea here is to make the `\item` command behave in the same way as in the `keyans` environment with the difference of the optional argument (`\number`) which works in the same way as in the `enumext*` environment. In simple terms we want to store the `\label` next to the `[\content]` if it is present in the `\sequence` and `\prop list` defined by `save-ans` key for `\item*`, `\item*[\content]`, `\item(\number)*` and `\item(\number)*[\content]` commands.

`__enumext_start_item_tmp_viii:` First we will call the function `__enumext_stop_item_tmp_viii:` that we will redefine later, we will increment the value of `\l__enumext_item_column_pos_viii_int` that will count the item's by rows and the value of `\g__enumext_item_count_all_viii_int` that will count the total of item's in the environment. After that we will call the function `__enumext_item_peek_args_viii:` that will handle the arguments passed to `\item`.

```

4470 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
4471 {
4472     \__enumext_stop_item_tmp_viii:
4473     \int_incr:N \l__enumext_item_column_pos_viii_int
4474     \int_gincr:N \g__enumext_item_count_all_viii_int
4475     \__enumext_item_peek_args_viii:
4476 }

```

(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)`.

```

4477 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
4478 {
4479     \peek_meaning:NTF (
4480         { \__enumext_joined_item_viii:w }
4481         { \__enumext_joined_item_viii:w (1) }
4482     )

```

(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`.

```

4483 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
4484 {
4485   \__enumext_starred_joined_item_viii:n {#1}
4486   \peek_meaning_remove:NTF *
4487   { \__enumext_starred_item_viii:w }
4488   { \__enumext_standar_item_viii:w }
4489 }

```

(End of definition for `__enumext_joined_item_viii:w`.)

`__enumext_standar_item_viii:w` The function `__enumext_standar_item_viii:w` will first look for the argument “[”, if present it will set the state of the variable `\l__enumext_wrap_label_opt_viii_bool` equal to the state of the variable `\l__enumext_wrap_label_opt_viii_bool` handled by the key `wrap-label*` and finally execute the *non-enumerated* version `\item[⟨custom⟩]` by means of the function `__enumext_start_item_viii:w`, otherwise we will set the value of the variable `\l__enumext_wrap_label_viii_bool` handled by the `wrap-label` key to true and set the switch `\if@noitemarg` to true to execute the enumerated version of `\item` by means of the function `__enumext_start_item_viii:w [\l__enumext_label_viii_tl]`.

```

4490 \cs_new_protected:Npn \__enumext_standar_item_viii:w
4491 {
4492   \bool_set_false:N \l__enumext_item_starred_viii_bool
4493   \peek_meaning:NTF [
4494   {
4495     \bool_set_eq:NN
4496     \l__enumext_wrap_label_viii_bool
4497     \l__enumext_wrap_label_opt_viii_bool
4498     \__enumext_start_item_viii:w
4499   }
4500   {
4501     \bool_set_true:N \l__enumext_wrap_label_viii_bool
4502     \legacy_if_set_true:n { @noitemarg }
4503     \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4504   }
4505 }

```

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

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

```

\__enumext_starred_item_viii_aux_i:w
\__enumext_starred_item_viii_aux_ii:w
4506 \cs_new_protected:Npn \__enumext_starred_item_viii:w
4507 {
4508   \bool_set_true:N \l__enumext_item_starred_viii_bool
4509   \bool_set_true:N \l__enumext_wrap_label_viii_bool
4510   \peek_meaning:NTF [
4511   { \__enumext_starred_item_viii_aux_i:w }
4512   { \__enumext_starred_item_viii_aux_ii:w }
4513 }

```

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`.

```

4514 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
4515 {
4516   \tl_clear:N \l__enumext_store_current_label_tl
4517   \tl_if_no_value:nF { #1 }
4518   {
4519     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
4520     {
4521       \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
4522       \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
4523     }
4524     \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
4525   }
4526   \__enumext_starred_item_viii_aux_ii:w
4527 }
4528 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w

```

```

4529 {
4530   \legacy_if_set_true:n { @noitemarg }
4531   \__enumext_start_item_viii:w [ \__enumext_label_viii_tl ]
4532 }

```

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

__enumext_starred_item_exec:

The function __enumext_starred_item_exec: will be in charge of storing the current *label* for \item* followed by the [*content*] for \item* [*content*] if present in the *sequence* and *prop list* set by the save-ans key. In this same function the keys show-ans, show-pos and save-ref are implemented.

```

4533 \cs_new_protected:Nn \__enumext_starred_item_exec:
4534 {
4535   \tl_put_left:Ne \__enumext_store_current_label_tl { \__enumext_label_viii_tl }
4536   \__enumext_store_addto_prop:V \__enumext_store_current_label_tl
4537   \__enumext_keyans_store_ref:
4538   \tl_put_left:Ne \__enumext_store_current_label_tl { \item }
4539   \__enumext_keyans_addto_seq_link:
4540   \int_gincr:N \g__enumext_check_starred_cmd_int
4541   \bool_if:NT \__enumext_show_answer_bool
4542   {
4543     \__enumext_print_keyans_box:NN \__enumext_labelwidth_i_dim \__enumext_labelsep_i_dim
4544   }
4545   \bool_if:NT \__enumext_show_position_bool
4546   {
4547     \tl_set:Ne \__enumext_mark_answer_sym_tl
4548     {
4549       \group_begin:
4550       \exp_not:N \normalfont
4551       \exp_not:N \footnotesize [ \int_eval:n
4552       {
4553         \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop }
4554       }
4555       ]
4556       \group_end:
4557     }
4558     \__enumext_print_keyans_box:NN \__enumext_labelwidth_i_dim \__enumext_labelsep_i_dim
4559   }
4560 }

```

(End of definition for __enumext_starred_item_exec:.)

12.44.2 Real definition of \item in keyans*

__enumext_start_item_viii:w

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

```

4561 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
4562 {
4563   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
4564   \legacy_if:nT { @noitemarg }
4565   {
4566     \legacy_if_set_false:n { @noitemarg }
4567     \legacy_if:nT { @nmbrrlist }
4568     {
4569       \bool_if:NT \__enumext_hyperref_bool
4570       {
4571         \legacy_if_set_true:n { @hyper@item }
4572       }
4573       \refstepcounter{enumXviii}
4574     }
4575   }

```

Here we start capturing \item and its *contents* in a *horizontal box*.

```

4576   \hbox_set_to_wd:Nnw \__enumext_item_text_viii_box
4577   {
4578     \__enumext_joined_width_viii_dim
4579     + \__enumext_labelwidth_viii_dim
4580     + \__enumext_labelsep_viii_dim
4581   }
4582   \bool_if:NF \__enumext_footnotes_key_bool
4583   {
4584     \__enumext_renew_footnote:
4585   }

```

```

4586     \bool_if:NT \l__enumext_item_starred_viii_bool
4587     {
4588         \__enumext_starred_item_exec:
4589     }
4590     \group_begin:
4591     \tl_use:N \l__enumext_label_font_style_viii_tl
4592     \bool_if:NTF \l__enumext_wrap_label_viii_bool
4593     {
4594         \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
4595         { \__enumext_wrapper_label_viii:n {#1} }
4596     }
4597     {
4598         \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1
4599     }
4600     \group_end:
4601     \skip_horizontal:N \l__enumext_labelsep_viii_dim
4602     \tl_use:N \l__enumext_after_list_args_viii_tl
4603     \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
4604     \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
4605     \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
4606     \bool_if:NT \l__enumext_item_starred_viii_bool
4607     {
4608         \tl_use:N \l__enumext_fake_item_indent_viii_tl
4609         \__enumext_keyans_show_item_opt:
4610         \skip_horizontal:n { -\l__enumext_fake_item_indent_viii_dim - \l__enumext_labelsep_viii_dim }
4611     }
4612     {
4613         \tl_use:N \l__enumext_fake_item_indent_viii_tl
4614     }
4615 }

```

(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.

```

4616 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
4617 {
4618     \__enumext_endminipage:
4619     \hbox_set_end:
4620     \int_set:Nn \hbadness { 10000 }
4621     \box_use_drop:N \l__enumext_item_text_viii_box
4622     \bool_if:NF \l__enumext_footnotes_key_bool
4623     {
4624         \__enumext_print_footnote:
4625     }
4626     \int_compare:nNnTF
4627     { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
4628     {
4629         \par\noindent
4630         \int_zero:N \l__enumext_item_column_pos_viii_int
4631     }
4632     {
4633         \skip_horizontal:N \l__enumext_columns_sep_viii_dim
4634     }
4635 }

```

(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.

```

4636 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
4637 {
4638     \int_compare:nNnT
4639     {
4640         \int_mod:nn
4641         { \g__enumext_item_count_all_viii_int }
4642         { \l__enumext_columns_viii_int }
4643     }
4644     =

```

```

4645     { 0 }
4646   {
4647     \para_end:
4648     \skip_vertical:n { -\__enumext_itemsep_viii_skip }
4649     \skip_vertical:N \c_zero_skip
4650     \int_gzero:N \g__enumext_item_count_all_viii_int
4651   }
4652 }

```

(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 $\langle \textit{store name} : \textit{position} \rangle$. Retrieve a “single” content stored by `\anskey`, `\anspic*` and `\item*` from $\langle \textit{prop list} \rangle$ defined by `save-ans` key.

```

4653 \NewDocumentCommand \getkeyans { m }
4654 {
4655   \exp_args:Ne \__enumext_getkeyans_aux:n
4656   { \tl_to_str:e { \text_expand:n {#1} } }
4657 }

```

(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 $\langle \textit{argument} \rangle$ using “.”. If “.” is omitted it will return an error.

```

4658 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
4659 {
4660   \str_if_in:nnTF {#1} { : }
4661   {
4662     \use:e
4663     {
4664       \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
4665       { {##1} {##2} }
4666     }
4667     \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
4668   }
4669   { \msg_error:nnn { enumext } { missing-colon } {#1} }
4670 }

```

(End of definition for __enumext_getkeyans_aux:n.)

`__enumext_getkeyans:nn` The internal function `__enumext_getkeyans:nn` will check for the existence of the $\langle \textit{prop list} \rangle$, if it does not exist it will return an error message, then it will fetch the content specified by the second $\langle \textit{argument} \rangle$ from $\langle \textit{prop list} \rangle$.

```

4671 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
4672 {
4673   \prop_if_exist:cTF { g__enumext_#1_prop }
4674   {
4675     \prop_item:cn { g__enumext_#1_prop }{#2}
4676   }
4677   {
4678     \msg_error:nnn { enumext } { undefined-storage-anskey } {#1}
4679   }
4680 }

```

(End of definition for __enumext_getkeyans:nn.)

12.46 The command \printkeyans

The `\printkeyans` command prints “all stored content” in the $\langle \textit{sequence} \rangle$ defined by the `save-ans` key. The first thing we will do is define a set of $\langle \textit{filtered keys} \rangle$ 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 $\langle \textit{keys} \rangle$ for `\printkeyans*` and will be set by `\setenumext[$\langle \textit{print}^* \rangle$]` and the variable `\l__enumext_print_keyans_vii_tl` will have the default keys for the environment `enumext*` nested within the $\langle \textit{sequence} \rangle$ and will be set by `\setenumext[$\langle \textit{print}^*, * \rangle$]`, the rest of the variables will be for the environment `enumext` and will be set by `\setenumext[$\langle \textit{print}^*, level \rangle$]`.

```

4681 \keys_define:nn { enumext / print }
4682 {
4683   print* .code:n = \keys_precompile:neN { enumext / enumext* }

```

```

4684         { \__enumext_filter_save_key:n {#1} }
4685         \__enumext_print_keyans_starred_tl, % starred cmd
4686     print* .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4687     print-1 .code:n = \keys_precompile:neN { enumext / level-1 }
4688         { \__enumext_filter_save_key:n {#1} }
4689         \__enumext_print_keyans_i_tl,
4690     print-1 .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4691     print-2 .code:n = \keys_precompile:neN { enumext / level-2 }
4692         { \__enumext_filter_save_key:n {#1} }
4693         \__enumext_print_keyans_ii_tl,
4694     print-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
4695     print-3 .code:n = \keys_precompile:neN { enumext / level-3 }
4696         { \__enumext_filter_save_key:n {#1} }
4697         \__enumext_print_keyans_iii_tl,
4698     print-3 .initial:n = { nosep, label=\roman*., first=\small, font=\small },
4699     print-4 .code:n = \keys_precompile:neN { enumext / level-4 }
4700         { \__enumext_filter_save_key:n {#1} }
4701         \__enumext_print_keyans_iv_tl,
4702     print-4 .initial:n = { nosep, label=\Alph*., first=\small, font=\small },
4703     print-* .code:n = \keys_precompile:neN { enumext / enumext* }
4704         { \__enumext_filter_save_key:n {#1} }
4705         \__enumext_print_keyans_vii_tl, % starred nested
4706     print-* .initial:n = { nosep, label=\arabic*., first=\small, font=\small },
4707 }

```

• 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`.

```

4708 \NewDocumentCommand \printkeyans { s O{} m }
4709 {
4710     \group_begin:
4711         \tl_use:N \__enumext_print_keyans_i_tl
4712         \tl_use:N \__enumext_print_keyans_ii_tl
4713         \tl_use:N \__enumext_print_keyans_iii_tl
4714         \tl_use:N \__enumext_print_keyans_iv_tl
4715         \tl_use:N \__enumext_print_keyans_vii_tl
4716         \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
4717     \group_end:
4718 }

```

(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.

```

4719 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
4720 {
4721     \seq_if_exist:cTF { g__enumext_#3_seq }
4722     {
4723         \seq_if_empty:cF { g__enumext_#3_seq }
4724         {
4725             %%\seq_show:c { g__enumext_#3_seq }

```

If the starred argument is present we will check that the environment `enumext*` is not saved in the `\sequence`, then execute the variable `__enumext_print_keyans_starred_tl` that contains the default `\keys` for the environment `enumext*`, it will open the environment `enumext*` passing the optional argument to the “first level”, set the key `base-fix` and then will map the `\sequence`.

```

4726         \bool_if:nTF {#1}
4727         {
4728             \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
4729             {
4730                 \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
4731             }
4732         {
4733             \tl_use:N \__enumext_print_keyans_starred_tl
4734             \begin{enumext*}[#2]
4735                 \keys_set:nn { enumext / level-1 } { base-fix }
4736                 \seq_map_inline:cn { g__enumext_#3_seq } { #1 }

```

```

4737         \end{enumext*}
4738     }
4739 }

```

Otherwise it will open the environment `enumext` passing the optional argument to the “*first level*”, set the key `base-fix` and then map the $\langle sequence \rangle$.

```

4740     {
4741         \begin{enumext} [#2]
4742             \keys_set:nn { enumext / enumext* } { base-fix }
4743             \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4744         \end{enumext}
4745     }
4746 }
4747 }
4748 {
4749     \msg_error:nnn { enumext } { undefined-storage-anskey } { #3 }
4750 }
4751 }

```

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

12.47 The command `\setenumext`

The command `\setenumext` will be in charge of managing the $\langle keys \rangle$ 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 $\langle keys \rangle$ that complicate us.

The function `__enumext_filter_first_level:n` will be in charge of filtering the $\langle keys \rangle$ passed to the environment `enumext*` and “*first level*” of the environment `enumext`.

```

4752 \cs_new:Npn \__enumext_filter_first_level:n #1
4753 {
4754     \use:e
4755     {
4756         \keyval_parse:NNn
4757         \__enumext_filter_first_level_key:n
4758         \__enumext_filter_first_level_pair:nn {#1}
4759     }
4760 }

```

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

```

4761 \cs_new:Npn \__enumext_filter_first_level_key:n #1
4762 {
4763     \str_case:nnF {#1}
4764     {
4765         { resume } {}
4766         { resume* } {}
4767     }
4768     { , { \exp_not:n {#1} } }
4769 }

```

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

```

4770 \cs_new:Npn \__enumext_filter_first_level_pair:nn #1#2
4771 {
4772     \str_case:nnF {#1}
4773     {
4774         { series } {}
4775         { resume } {}
4776         { save-ans } {}
4777     }
4778     { , { \exp_not:n {#1} } } = { \exp_not:n {#2} } }
4779 }

```

(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 $\langle keys \rangle$ to access from `\setenumext`.

```

4780 \keys_define:nn { enumext / meta-families }
4781 {
4782     enumext-1 .code:n =
4783     {
4784         \keys_set:ne { enumext / level-1 }
4785         {

```



```

4786         \__enumext_filter_first_level:n {#1}
4787     }
4788 },
4789 enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
4790 enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
4791 enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
4792 keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} } ,
4793 enumext* .code:n =
4794 {
4795     \keys_set:ne { enumext / enumext* }
4796     {
4797         \__enumext_filter_first_level:n {#1}
4798     }
4799 },
4800 keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
4801 print* .code:n = { \keys_set:nn { enumext / print } { print* = {#1} } } ,
4802 print-1 .code:n = { \keys_set:nn { enumext / print } { print-1 = {#1} } } ,
4803 print-2 .code:n = { \keys_set:nn { enumext / print } { print-2 = {#1} } } ,
4804 print-3 .code:n = { \keys_set:nn { enumext / print } { print-3 = {#1} } } ,
4805 print-4 .code:n = { \keys_set:nn { enumext / print } { print-4 = {#1} } } ,
4806 print-* .code:n = { \keys_set:nn { enumext / print } { print-* = {#1} } } ,
4807 unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
4808 }

```

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

```

4809 \seq_const_from_clist:Nn \c__enumext_all_families_seq
4810 {
4811     enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
4812     keyans*, print-1, print-2, print-3, print-4, print-*, print*,
4813 }

```

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

```

4814 \NewDocumentCommand \setenumext { 0{enumext,1} +m }
4815 {
4816     \seq_clear:N \__enumext_setkey_tmpa_seq
4817     \seq_set_from_clist:Nn \__enumext_setkey_tmpb_seq {#1}
4818     \int_set:Nn \__enumext_setkey_tmpa_int
4819     {
4820         \seq_count:N \__enumext_setkey_tmpb_seq
4821     }
4822     \int_compare:nNnTF { \__enumext_setkey_tmpa_int } > { 1 }
4823     {
4824         \seq_pop_left:NN \__enumext_setkey_tmpb_seq \__enumext_setkey_tmpa_tl
4825         \seq_map_function:NN \__enumext_setkey_tmpb_seq \__enumext_set_parse:n
4826         \seq_set_map_e:NNn \__enumext_setkey_tmpa_seq \__enumext_setkey_tmpa_seq
4827         {
4828             \tl_use:N \__enumext_setkey_tmpa_tl - ##1
4829         }
4830     }
4831     {
4832         \seq_put_right:Ne \__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
4833     }
4834     \seq_if_empty:NNTF \__enumext_setkey_tmpa_seq
4835     { \seq_map_inline:Nn \c__enumext_all_families_seq }
4836     { \seq_map_inline:Nn \__enumext_setkey_tmpa_seq }
4837     {
4838         \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
4839     }
4840 }

```

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

`__enumext_set_parse:n`
`__enumext_set_error:n`

Internal functions used by the `\setenumext` command.

```

4841 \cs_new_protected:Npn \__enumext_set_parse:n #1
4842 {
4843     \tl_set:Ne \__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
4844     \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
4845     { \tl_remove_all:Nn \__enumext_setkey_tmpb_tl {##1} }
4846     \tl_if_empty:NNTF \__enumext_setkey_tmpb_tl
4847     {
4848         \seq_put_right:Ne \__enumext_setkey_tmpa_seq

```

```

4849         { \tl_trim_spaces:n {#1} }
4850     }
4851     { \__enumext_set_error:nn {#1} { } }
4852 }
4853 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
4854 { \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

```

```

4855 \prop_const_from_keyval:Nn \c__enumext_meta_paths_prop
4856 {
4857     {enumext,1} = level-1,
4858     {enumext,2} = level-2,
4859     {enumext,3} = level-3,
4860     {enumext,4} = level-4,
4861     {enumext*} = enumext*
4862 }

```

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

```

4863 \NewDocumentCommand \setenumextmeta { s O{enumext,1} m +m }
4864 {
4865     \str_if_eq:eeTF { \tl_trim_spaces:n {#3} } { unknown }
4866     { \msg_error:nn { enumext } { prohibited-unknown } }
4867     {
4868         \bool_if:nTF {#1}
4869         {
4870             \int_step_inline:nn { 4 }
4871             { \__enumext_add_meta_key:nnn { enumext, ##1 } {#3} {#4} }
4872             \__enumext_add_meta_key:nnn { enumext* } {#3} {#4}
4873         }
4874         { \__enumext_add_meta_key:nnn {#2} {#3} {#4} }
4875     }
4876 }

```

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

```

4877 \cs_new_protected:Npn \__enumext_add_meta_key:nnn #1
4878 {
4879     \tl_set:Nn \l__enumext_meta_path_tl {#1}
4880     \tl_replace_all:Nnn \l__enumext_meta_path_tl { ~ } {}
4881     \prop_get:NVNTF
4882     \c__enumext_meta_paths_prop \l__enumext_meta_path_tl \l__enumext_meta_path_tl
4883     { \__enumext_def_meta_key:Vnn \l__enumext_meta_path_tl }
4884     {
4885         \msg_error:nnn { enumext } { unknown-set } {#1}
4886         \use_none:n
4887     }
4888 }
4889 \cs_new_protected:Npn \__enumext_def_meta_key:nnn #1#2#3
4890 {
4891     \bool_lazy_or:nnTF
4892     { \keys_if_exist_p:nn { enumext / #1 } {#2} }
4893     { \keys_if_exist_p:nn { enumext / enumext* } {#2} }
4894     { \msg_error:nnn { enumext } { already-defined } {#2} }
4895     {
4896         \keys_define:nn { enumext / #1 }
4897         {
4898             #2 .meta:n = {#3},
4899             #2 .value_forbidden:n = true
4900         }
4901     }
4902 }
4903 \cs_generate_variant:Nn \__enumext_def_meta_key:nnn { V }

```

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

12.49 The command \foreachkeyans

The command `\foreachkeyans` will execute a *loop* over the `<prop list>` and return its contents. The implementation code is adapted from the answer provided by Enrico Gregorio (@egreg) in [Expand a .cs defined by key inside the function.](#)

`\foreachkeyans`

`__enumext_parse_foreach_keys:nn`

`__enumext_parse_foreach_keys:n`

`__enumext_foreach_keyans:nn`

`__enumext_foreach_add_body:n`

We define a set of `<keys>` for command and we will save the default values of these in `\g__enumext_foreach_default_keys_tl` to avoid the use of group.

```

4904 \keys_define:nn { enumext / foreach }
4905 {
4906   before .tl_set:N = \l__enumext_foreach_before_tl,
4907   before .value_required:n = true,
4908   after .tl_set:N = \l__enumext_foreach_after_tl,
4909   after .value_required:n = true,
4910   start .int_set:N = \l__enumext_foreach_start_int,
4911   start .value_required:n = true,
4912   stop .int_set:N = \l__enumext_foreach_stop_int,
4913   stop .value_required:n = true,
4914   step .int_set:N = \l__enumext_foreach_step_int,
4915   step .value_required:n = true,
4916   wrapper .cs_set_protected:Np = \__enumext_foreach_wrapper:n #1,
4917   wrapper .value_required:n = true,
4918   sep .tl_set:N = \l__enumext_foreach_sep_tl,
4919   sep .value_required:n = true,
4920   unknown .code:n = { \__enumext_parse_foreach_keys:n {#1} }
4921 }
4922 \keys_precompile:nnN { enumext / foreach }
4923 {
4924   before={},after={},start=1,step=1,stop=0,wrapper=#1,sep=
4925 }
4926 \g__enumext_foreach_default_keys_tl

```

Functions for handling unknown `<keys>`.

```

4927 \cs_new_protected:Npn \__enumext_parse_foreach_keys:nn #1#2
4928 {
4929   \tl_if_blank:nTF {#2}
4930   {
4931     \msg_error:nnn { enumext } { for-key-unknown } {#1}
4932   }
4933   {
4934     \msg_error:nnnn { enumext } { for-key-value-unknown } {#1} {#2}
4935   }
4936 }
4937 \cs_new_protected:Npn \__enumext_parse_foreach_keys:n #1
4938 {
4939   \exp_args:NV \__enumext_parse_foreach_keys:nn \l_keys_key_str {#1}
4940 }

```

We create the command.

```

4941 \NewDocumentCommand \foreachkeyans { +0{} m }
4942 {
4943   \__enumext_foreach_keyans:nn {#1} {#2}
4944 }

```

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

```

4945 \cs_new_protected:Npn \__enumext_foreach_keyans:nn #1 #2
4946 {
4947   \tl_use:N \g__enumext_foreach_default_keys_tl
4948   \keys_set:nn { enumext / foreach } {#1}
4949   \tl_set:Nn \l__enumext_foreach_name_prop_tl {#2}
4950   \prop_if_exist:cF { g__enumext_#2_prop }
4951   {
4952     \msg_error:nnn { enumext } { undefined-storage-anskey } {#2}
4953   }
4954   \int_compare:nNnT { \l__enumext_foreach_stop_int } = { 0 }
4955   {
4956     \int_set:Nn \l__enumext_foreach_stop_int
4957     { \prop_count:c { g__enumext_#2_prop } }
4958   }
4959   \seq_clear:N \l__enumext_foreach_print_seq
4960   \int_step_function:nnnN

```

```

4961     { \l__enumext_foreach_start_int }
4962     { \l__enumext_foreach_step_int }
4963     { \l__enumext_foreach_stop_int }
4964     \__enumext_foreach_add_body:n
4965     \seq_use:NV \l__enumext_foreach_print_seq \l__enumext_foreach_sep_tl
4966   }
4967   \cs_new_protected:Npn \__enumext_foreach_add_body:n #1
4968   {
4969     \seq_put_right:Ne \l__enumext_foreach_print_seq
4970     {
4971       \exp_not:V \l__enumext_foreach_before_tl
4972       \__enumext_foreach_wrapper:n
4973       {
4974         \prop_item:cn { g__enumext_ \l__enumext_foreach_name_prop_tl _prop }{#1}
4975       }
4976       \exp_not:V \l__enumext_foreach_after_tl
4977     }
4978   }

```

(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.

```

4979 \msg_new:nnn { enumext } { package-load }
4980 {
4981   The ~ '#1' ~ package ~ is ~ already ~ loaded.
4982 }
4983 \msg_new:nnn { enumext } { package-not-load }
4984 {
4985   The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
4986 }
4987 \msg_new:nnn { enumext } { package-load-foot }
4988 {
4989   The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
4990 }

```

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

```

4991 \msg_new:nnn { enumext } { counters }
4992 {
4993   The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
4994   package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
4995 }

```

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

```

4996 \msg_new:nnn { enumext } { unknown-choice }
4997 {
4998   The ~ value ~ '#3' ~ for ~ '#1' ~ key ~ is ~ invalid ~ use ~ ('#2').
4999 }

```

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

```

5000 \msg_new:nnnn { enumext } { anskey-env-error }
5001 {
5002   The ~ '#1' ~ environment ~is~ reserved ~ by ~\\
5003   'enumext' ~ package, ~ It~ is~ already~ defined.
5004 }
5005 {
5006   The ~ anskey* ~ environment ~ is ~ defined ~ internally ~
5007   for ~ the ~ 'save-ans' ~ key.\\
5008 }

```

Message used in the creation of `⟨prop list⟩` by `enumext` package.

```

5009 \msg_new:nnn { enumext } { store-prop }
5010 {
5011   * ~ Package ~ enumext: ~ Creating ~
5012   \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
5013 }
5014 \msg_new:nnn { enumext } { store-seq }
5015 {
5016   * ~ Package ~ enumext: ~ Creating ~
5017   \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
5018 }
5019 \msg_new:nnn { enumext } { store-int }

```

```

5020 {
5021   * ~ Package ~ enumext: ~ Creating ~
5022   \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_context:.
5023 }
5024 \msg_new:nnn { enumext } { prop-seq-int-hook }
5025 {
5026   * ~ Package ~ enumext: ~ Elements ~ in ~
5027   \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\
5028   * ~ Package ~ enumext: ~ Elements ~ in ~
5029   \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\
5030   * ~ Package ~ enumext: ~ Value ~ off ~
5031   \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
5032 }
5033 \msg_new:nnn { enumext } { item-answer-hook }
5034 {
5035   * ~ Package ~ enumext: ~ Value ~ off ~
5036   \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\
5037   * ~ Package ~ enumext: ~ Value ~ off ~
5038   \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\
5039   * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
5040 }

```

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

```

5041 \msg_new:nnn { enumext } { invalid-key }
5042 {
5043   The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
5044 }
5045 \msg_new:nnn { enumext } { unknown-key-family }
5046 {
5047   Unknown~key~family~`\l_keys_key_str'~for~enumext.
5048 }

```

Messages used in length calculation.

```

5049 \msg_new:nnn { enumext } { width-negative }
5050 {
5051   Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
5052   The ~ key ~ '#1'~ accepts ~ values ~ >= ~ 0pt.
5053 }
5054 \msg_new:nnn { enumext } { width-zero }
5055 {
5056   Invalid ~ '#1=#2' ~ \msg_line_context:.\
5057   The ~ key ~ '#1'~ accepts ~ values ~ > ~ 0pt.
5058 }

```

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

```

5059 \msg_new:nnn { enumext } { list-lengths }
5060 {
5061   **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\
5062   \__enumext_show_length:nnn { dim } { labelsep } {#1}
5063   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
5064   \__enumext_show_length:nnn { dim } { itemindent } {#1}
5065   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
5066   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
5067   \__enumext_show_length:nnn { dim } { listparindent } {#1}
5068   \__enumext_show_length:nnn { skip } { topsep } {#1}
5069   \__enumext_show_length:nnn { skip } { parsep } {#1}
5070   \__enumext_show_length:nnn { skip } { partopsep } {#1}
5071   \__enumext_show_length:nnn { skip } { itemsep } {#1}
5072   ****
5073 }

```

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

```

5074 \msg_new:nnn { enumext } { list-lengths-not-nested }
5075 {
5076   **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\
5077   \__enumext_show_length:nnn { dim } { labelsep } {#1}
5078   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
5079   \__enumext_show_length:nnn { dim } { itemindent } {#1}
5080   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
5081   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
5082   \__enumext_show_length:nnn { dim } { listparindent } {#1}
5083   \__enumext_show_length:nnn { skip } { topsep } {#1}
5084   \__enumext_show_length:nnn { skip } { parsep } {#1}

```

```

5085     \__enumext_show_length:nnn { skip } { partopsep } {#1}
5086     \__enumext_show_length:nnn { skip } { itemsep } {#1}
5087     *****
5088 }

```

Messages used by `ref` key.

```

5089 \msg_new:nnn { enumext } { key-ref-empty }
5090 {
5091     Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
5092 }

```

Messages used by `save-ans` key.

```

5093 \msg_new:nnn { enumext } { save-ans-empty }
5094 {
5095     Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
5096 }
5097 \msg_new:nnn { enumext } { save-ans-log }
5098 {
5099     * ~ Package ~ enumext: ~ Start ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5100 }
5101 \msg_new:nnn { enumext } { save-ans-log-hook }
5102 {
5103     * ~ Package ~ enumext: ~ Stop ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5104 }
5105 \msg_new:nnn { enumext } { save-ans-hook }
5106 {
5107     Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
5108 }

```

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

```

5109 \msg_new:nnn { enumext } { need-save-ans }
5110 {
5111     Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
5112 }
5113 \msg_new:nnn { enumext } { items-same-answer }
5114 {
5115     *****\
5116     * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~
5117     for ~ \c_left_brace_str #2 \c_right_brace_str\
5118     * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~
5119     'OK', ~ all ~ items ~ with ~ answer.\
5120     *****
5121 }
5122 \msg_new:nnn { enumext } { item-greater-answer }
5123 {
5124     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\
5125     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\
5126     Items ~ > ~ Answers.
5127 }
5128 \msg_new:nnn { enumext } { item-less-answer }
5129 {
5130     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\
5131     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\
5132     Items ~ < ~ Answers.
5133 }

```

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

```

5134 \msg_new:nnn { enumext } { missing-starred }
5135 {
5136     Missing ~ '\c_backslash_str #1*' ~ #2.
5137 }
5138 \msg_new:nnn { enumext } { many-starred }
5139 {
5140     Many ~ '\c_backslash_str #1*' ~ #2.
5141 }

```

Messages used by `\printkeyans*` command.

```

5142 \msg_new:nnn { enumext } { print-starred }
5143 {
5144     \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
5145     #2 ~ environment ~ \msg_line_context:.
5146 }

```

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

```
5147 \msg_new:nnn { enumext } { list-too-deep }
5148 {
5149     Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \\
5150     The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
5151 }
```

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

```
5152 \msg_new:nnn { enumext } { anskey-unnumber-item }
5153 {
5154     Can't ~ store ~ with ~ a ~ unnumbered ~ \c_backslash_str item ~ \msg_line_context:.
5155 }
5156 \msg_new:nnn { enumext } { anskey-already-stored }
5157 {
5158     Content ~ already ~ stored ~ for ~ this ~ \c_backslash_str item ~ \msg_line_context:.
5159 }
5160 \msg_new:nnn { enumext } { anskey-empty-arg }
5161 {
5162     Can't ~ store ~ empty ~ content ~ \msg_line_context:.
5163 }
5164 \msg_new:nnn { enumext } { anskey-wrong-place }
5165 {
5166     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5167     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5168 }
5169 \msg_new:nnn { enumext } { anskey-nested }
5170 {
5171     The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
5172 }
5173 \msg_new:nnn { enumext } { anskey-math-mode }
5174 {
5175     #1 ~ can't ~ work ~ in ~ math ~ mode ~ \msg_line_context:.
5176 }
5177 \msg_new:nnn { enumext } { anskey-env-wrong }
5178 {
5179     The ~ environment ~ anskey* ~ cannot ~ use ~ in ~ '#1' ~ \msg_line_context:.
5180 }
5181 \msg_new:nnn { enumext } { anspic-wrong-place }
5182 {
5183     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5184     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5185 }
5186 \msg_new:nnn { enumext } { command-wrong-place }
5187 {
5188     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5189     '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
5190 }
5191 \msg_new:nnnn { enumext } { anskey-env-key-unknown }
5192 {
5193     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5194     'anskey*' ~ and ~ is ~ being ~ ignored.
5195 }
5196 {
5197     The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~ '#1'.\\
5198     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5199 }
5200 \msg_new:nnnn { enumext } { anskey-env-key-value-unknown }
5201 {
5202     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5203     'anskey*' ~ and ~ is ~ being ~ ignored.
5204 }
5205 {
5206     The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~ '#1'.\\
5207     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5208 }
5209 \msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
5210 { The ~ key ~ '#1' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.}
5211 {
5212     The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~ '#1'.\\
5213     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5214 }
```



```

5215 \msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
5216 { The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.
5217 {
5218   The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5219   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5220 }

```

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

```

5221 \msg_new:nnn { enumext } { keyans-nested }
5222 {
5223   The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
5224 }
5225 \msg_new:nnn { enumext } { keyans-wrong-level }
5226 {
5227   Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
5228   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5229 }
5230 \msg_new:nnn { enumext } { wrong-place }
5231 {
5232   Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:~ \\
5233   '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext'.
5234 }
5235 \msg_new:nnn { enumext } { keyanspic-nested }
5236 {
5237   The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested ~ \msg_line_context:~.
5238 }
5239 \msg_new:nnn { enumext } { keyanspic-wrong-level }
5240 {
5241   Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
5242   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5243 }
5244 \msg_new:nnn { enumext } { keyanspic-item-cmd }
5245 {
5246   Can't ~ use ~ \c_backslash_str item ~ in ~ keyanspic ~ \msg_line_context:.
5247 }
5248 \msg_new:nnnn { enumext } { keyans-unknown-key }
5249 {
5250   The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5251   '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5252 }
5253 {
5254   The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5255   ~ have ~ a ~ key ~ called ~'#1'.\\
5256   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5257 }
5258 \msg_new:nnnn { enumext } { keyans-unknown-key-value }
5259 {
5260   The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5261   '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5262 }
5263 {
5264   The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5265   ~ have ~ a ~ key ~ called ~'#1'.\\
5266   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5267 }

```

Message used by unknown `<keys>` in `enumext*` environment.

```

5268 \msg_new:nnnn { enumext } { starred-unknown-key }
5269 {
5270   The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5271   '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5272 }
5273 {
5274   The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5275   ~ have ~ a ~ key ~ called ~'#1'.\\
5276   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5277 }
5278 \msg_new:nnnn { enumext } { starred-unknown-key-value }
5279 {
5280   The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5281   '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5282 }

```

```

5283 {
5284     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5285     ~ have ~ a ~ key ~ called ~ '#1'.\\
5286     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5287 }

```

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

```

5288 \msg_new:nnnn { enumext } { standar-unknown-key }
5289 {
5290     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_tl
5291     ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5292 }
5293 {
5294     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5295     ~ have ~ a ~ key ~ called ~ '#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5296     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5297 }
5298 \msg_new:nnnn { enumext } { standar-unknown-key-value }
5299 {
5300     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_
5301     ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5302 }
5303 {
5304     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5305     ~ have ~ a ~ key ~ called ~ '#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5306     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5307 }

```

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

```

5308 \msg_new:nnnn { enumext } { for-key-unknown }
5309 { The~key~'#1'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored.}
5310 {
5311     The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5312     Check~that~you~have~spelled~the~key~name~correctly.
5313 }
5314 \msg_new:nnnn { enumext } { for-key-value-unknown }
5315 { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored. }
5316 {
5317     The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5318     Check~that~you~have~spelled~the~key~name~correctly.
5319 }

```

Messages used by \getkeyans command.

```

5320 \msg_new:nnn { enumext } { undefined-storage-anskey }
5321 {
5322     Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
5323 }

```

Messages used by \miniright command.

```

5324 \msg_new:nnn { enumext } { missing-miniright }
5325 {
5326     Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\\
5327     The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
5328 }
5329 \msg_new:nnn { enumext } { wrong-miniright-place }
5330 {
5331     Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
5332     Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
5333 }
5334 \msg_new:nnn { enumext } { wrong-miniright-use }
5335 {
5336     Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
5337     '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
5338 }
5339 \msg_new:nnn { enumext } { wrong-miniright-starred }
5340 {
5341     Can't ~ use ~ \c_backslash_str miniright ~ in ~ starred ~ environments ~ \msg_line_context:.
5342 }
5343 \msg_new:nnn { enumext } { many-miniright-used }
5344 {
5345     Can't ~ use ~ \c_backslash_str miniright ~ more ~ than ~ once ~ \msg_line_context:.
5346 }

```

Messages used by `\setenumextmeta` command.

```

5347 \msg_new:nnn { enumext } { unknown-set }
5348 {
5349   Argument ~ [#1] ~ is ~ unknown ~ by ~ \c_backslash_str setenumextmeta ~ \msg_line_context:.
5350 }
5351 \msg_new:nnn { enumext } { already-defined }
5352 {
5353   The ~ key ~ '#1' ~ is ~ already ~ defined ~ \msg_line_context:.
5354 }
5355 \msg_new:nnn { enumext } { prohibited-unknown }
5356 {
5357   The ~ name ~ 'unknown' ~ can't ~ be ~ chosen~ for ~ a ~ meta ~ key ~ \msg_line_context:.
5358 }

```

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

```

5359 \msg_new:nnn { enumext } { nested }
5360 {
5361   The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ \msg_line_context:.
5362 }
5363 \msg_new:nnn { enumext } { nested-horizontal }
5364 {
5365   The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ in ~ '#1' ~ '
5366 }
5367 \msg_new:nnn { enumext } { item-joined }
5368 {
5369   Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~ \msg_line_context:.
5370 }
5371 \msg_new:nnn { enumext } { item-joined-columns }
5372 {
5373   Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~ \msg_line_context:.
5374 }

```

12.51 Finish package

Finish package implementation.

```

5375 \file_input_stop:
5376 </package>

```

13 Index of Implementation

The *italic* numbers denote the pages where the corresponding entry is described, the numbers underlined and all others indicate the line on which they are implemented in the package code.

Symbols	
<code>*</code>	219
<code>\+</code>	211
<code>\-</code>	211
<code>\\</code>	227, 2756, 3748, 4993, 5002, 5007, 5027, 5029, 5036, 5038, 5051, 5056, 5061, 5076, 5115, 5117, 5119, 5124, 5125, 5130, 5131, 5149, 5166, 5183, 5188, 5197, 5206, 5212, 5218, 5227, 5232, 5241, 5255, 5265, 5275, 5285, 5295, 5305, 5311, 5317, 5326, 5331, 5336
A	
<code>above</code>	<u>1571</u>
<code>above*</code>	<u>1571</u>
<code>\addvspace</code>	1140, 1169, 1212, 1215, 1383, 1386, 1483, 1489, 1524, 1530, 1551, 1557, 3567, 3702, 3717, 4075, 4089, 4130, 4144
<code>after</code>	<u>978</u>
<code>align</code>	<u>527</u>
<code>\Alph</code>	36, <u>41</u>
<code>\Alph</code>	479, 594, 639, 707, 4702
<code>\alph</code>	36, <u>41</u>
<code>\alph</code>	480, 592, 4694
<code>\anskey</code>	12, 74, 76, <u>2574</u>
<code>anskey*</code>	13, <u>2684</u>
<code>\anspic</code>	15, <u>100</u> , <u>3726</u>
<code>\anspic*</code>	68
<code>\arabic</code>	30, <u>36</u>
<code>\arabic</code>	478, 591, 638, 4686, 4690, 4706
B	
<code>base-fix</code>	<u>837</u>
<code>\baselineskip</code>	<u>50</u>
<code>\baselineskip</code>	854, <u>865</u>
<code>before</code>	<u>978</u>
<code>before*</code>	<u>978</u>
<code>below</code>	<u>1571</u>
<code>below*</code>	<u>1571</u>
bool commands:	
<code>\bool_gset_false:N</code>	350, 351, 352, 2860, 2862, 4091, 4095, 4146
<code>\bool_gset_true:N</code>	258, 268, 1081, 2064, 2070, 4067, 4092, 4122, 4147
<code>\bool_if:NTF</code>	418, 430, 447, 1505, 1593, 1607, 1620, 1631, 1642, 1653, 1664, 1675, 1724, 1741, 1746, 1754, 1781, 1819, 1824, 1831, 1835, 1857, 1862, 1870, 1877, 1908, 1916, 2009, 2207, 2217, 2296, 2320, 2327, 2351, 2449, 2471, 2511, 2524, 2528, 2578, 2597, 2621, 2675, 2686, 2775, 2812, 2876, 2909, 2924, 2999, 3010, 3014, 3033, 3046, 3088, 3122, 3157, 3291, 3353, 3363, 3395, 3400, 3503, 3550, 3572, 3631, 3686, 3707, 3728, 4063, 4072, 4076, 4118, 4127, 4131, 4221, 4231, 4314, 4319, 4332, 4336, 4351, 4373, 4429, 4439, 4541, 4545, 4569, 4582, 4586, 4592, 4606, 4622
<code>\bool_if:nTF</code>	1531, 1558, 3144, 3273, 3311, 3749, 4726, 4868
<code>\bool_if_p:N</code>	277, 292, 850, 851, 861, 862, 1888, 1889, 1897, 1898, 2022, 2048, 2061, 2062, 2067, 2068, 2384, 2394, 2406, 2421, 2422, 2456, 2497, 2498, 2798, 2986, 2987, 3024, 3025, 3476, 3478, 3489, 3756, 3757
<code>\bool_lazy_all:nTF</code>	275, 290, 2020, 2046, 2382, 2391, 2404, 2419, 3474, 3487
<code>\bool_lazy_and:nnTF</code>	254, 264, 849, 860, 1498, 1887, 1896, 2060, 2066, 2455, 2462, 2496, 2639, 2651, 2797, 2803, 2985
<code>\bool_lazy_or:nnTF</code>	1949, 1956, 3023, 3755, 4891
<code>\bool_new:N</code>	34, 35, 36, 37, 38, 39, 40, 41, 64, 73, 95, 100, 101, 106, 107, 110, 135, 136, 144, 145, 150, 152, 153, 167, 179, 181
<code>\bool_not_p:n</code>	255, 265, 2393, 2457, 2463, 2799, 2804, 3477, 3490
<code>\bool_set_eq:NN</code>	3097, 3253, 4264, 4495
<code>\bool_set_false:N</code>	427, 871, 1994, 1995, 2027, 2032, 2036, 2040, 2053, 2739, 3451, 3589, 3639, 3722, 3787, 3805, 4189, 4216, 4261, 4445, 4492
<code>\bool_set_true:N</code>	282, 283, 297, 298, 409, 413, 520, 886, 1577, 1582, 1844, 1966, 1967, 2239, 2247, 2740, 3091, 3093, 3125, 3127, 3249, 3261, 3375, 3450, 3483, 3496, 3522, 3636, 3663, 4052, 4107, 4188, 4270, 4277, 4278, 4322, 4443, 4501, 4508, 4509
box commands:	
<code>\box_dp:N</code>	1429, 1430, 1433, 1440, 1453, 1461, 1467, 1475, 3817
<code>\box_ht:N</code>	1212, 1215, 1226, 1227, 1238, 1240, 1255, 1258, 1266, 1267, 1278, 1280, 1295, 1298, 1305, 1306, 1317, 1319, 1334, 1337, 1383, 1386, 1394, 1395, 1403, 1404, 1416, 1418
<code>\box_new:N</code>	70, 174, 180
<code>\box_use_drop:N</code>	4087, 4142, 4372, 4621
<code>\box_wd:N</code>	486
C	
<code>\c</code>	219, 220, 744, 746, 758, 760
<code>\catcode</code>	2756
<code>\cB</code>	220
<code>\cE</code>	220
<code>\centering</code>	1533, 1560, 3843, 4080, 4135
<code>check-ans</code>	<u>1986</u>
Document class:	
<code>article</code>	43
clist commands:	
<code>\clist_const:Nn</code>	186
<code>\clist_map_function:nN</code>	3830
<code>\clist_map_inline:Nn</code>	526, 792, 977, 992, 1073, 1587
<code>\clist_map_inline:nn</code>	49, 60, 78, 85, 97, 109, 138, 161, 185, 554, 574, 846, 891, 912, 1087, 1693, 1933, 2000, 2186, 2204, 2236, 2379, 2918, 3178, 3190, 3230, 3340, 3343, 3370, 3382, 3385, 3405, 4844
<code>\columnbreak</code>	75
<code>\columnbreak</code>	2459
<code>columns</code>	<u>1057</u>
<code>columns-sep</code>	<u>1057</u>
<code>\columnsep</code>	96
<code>\columnsep</code>	3545, 3684
<code>\columnseprule</code>	96
<code>\columnseprule</code>	3548, 3685
Commands provide by enumext :	
<code>\anskey</code>	28, 65, 70–74, 76, 77, 83, 85, 95, 110, 119, 120, 128
<code>\anspic*</code>	28, 29, 68, 71, 83, 84, 100, 101, 103, 119, 120
<code>\anspic</code>	72, 100–102, 128

<code>\foreachkeyans</code>	124, 130
<code>\getkeyans</code>	71, 119, 130
<code>\item*</code> 28, 29, 68, 71, 72, 83, 84, 87, 90, 111, 116, 117, 119, 120	
<code>\item</code>	87, 90, 105, 110–112, 115, 116
<code>\miniright</code>	27, 47, 55, 56, 96, 97, 130
<code>\printkeyans*</code>	119
<code>\printkeyans</code>	28, 72, 119, 120
<code>\setenumextmeta</code>	123, 131
<code>\setenumext</code>	28, 120–122, 126
Counters defined by enumext :	
<code>enumXiii</code>	26, 36
<code>enumXii</code>	26, 36
<code>enumXiv</code>	26, 36
<code>enumXi</code>	26, 36
<code>enumXviii</code>	26, 36
<code>enumXvii</code>	26, 36, 112
<code>enumXvi</code>	26, 36
<code>enumXv</code>	26, 36
cs commands:	
<code>\cs_generate_variant:Nn</code> . 191, 192, 488, 504, 750, 766, 2288, 2293, 2369, 2692, 3330, 3832, 4903	
<code>\cs_if_exist:NTF</code>	458
<code>\cs_if_free:NTF</code>	2643, 2655
<code>\cs_new:Nn</code>	205
<code>\cs_new:Npn</code> . 223, 1694, 1703, 1711, 2251, 2260, 2268, 4752, 4761, 4770	
<code>\cs_new_eq:NN</code> 377, 378, 379, 383, 384, 432, 433, 436, 437	
<code>\cs_new_protected:Nn</code> . 215, 247, 273, 306, 336, 342, 348, 354, 360, 368, 386, 404, 615, 678, 730, 847, 993, 997, 1001, 1005, 1009, 1013, 1017, 1021, 1025, 1029, 1033, 1037, 1041, 1045, 1049, 1053, 1088, 1100, 1124, 1142, 1153, 1171, 1197, 1218, 1343, 1369, 1389, 1422, 1444, 1479, 1485, 1588, 1602, 1616, 1627, 1638, 1649, 1660, 1671, 1752, 1855, 1868, 1885, 1906, 1934, 1939, 1964, 2005, 2015, 2058, 2073, 2080, 2089, 2094, 2099, 2104, 2113, 2118, 2123, 2294, 2318, 2325, 2349, 2356, 2370, 2595, 2614, 2630, 2693, 2729, 2760, 2795, 2837, 2858, 2866, 2907, 2922, 2950, 2983, 3019, 3031, 3044, 3130, 3140, 3151, 3269, 3285, 3426, 3443, 3472, 3501, 3508, 3529, 3559, 3570, 3612, 3629, 3653, 3670, 3695, 3705, 3745, 3789, 3803, 3828, 3833, 3849, 3853, 3872, 3882, 3913, 4042, 4061, 4097, 4116, 4175, 4203, 4210, 4219, 4229, 4246, 4387, 4427, 4458, 4464, 4477, 4533, 4636	
<code>\cs_new_protected:Npn</code> 193, 197, 201, 229, 440, 456, 473, 483, 489, 595, 640, 712, 737, 751, 1515, 1544, 1720, 1739, 1809, 1842, 1944, 2128, 2205, 2215, 2237, 2245, 2280, 2289, 2445, 2508, 2522, 2560, 2564, 2684, 2715, 2719, 2750, 2886, 2960, 3004, 3084, 3103, 3191, 3195, 3209, 3213, 3231, 3235, 3245, 3257, 3299, 3333, 3373, 3454, 3649, 3798, 3944, 3993, 4192, 4252, 4259, 4275, 4283, 4288, 4300, 4451, 4483, 4490, 4506, 4514, 4528, 4658, 4671, 4719, 4841, 4853, 4877, 4889, 4927, 4937, 4945, 4967	
<code>\cs_new_protected_nopar:Nn</code> . . . 4239, 4367, 4470, 4616	
<code>\cs_new_protected_nopar:Npn</code>	4306, 4561
<code>\cs_set:Npn</code>	2380, 2417, 4664
<code>\cs_set_eq:NN</code> . . 4164, 4165, 4308, 4416, 4417, 4563	
<code>\cs_set_protected:Nn</code>	917, 933, 945, 957
<code>\cs_set_protected:Npn</code> . 45, 54, 71, 79, 92, 98, 131, 157, 165, 505, 527, 559, 575, 622, 767, 793, 837, 873, 896, 969, 978, 1057, 1074, 1571, 1682, 1925, 1986, 2145, 2187, 2223, 2372, 2911, 3167, 3183, 3223, 3331, 3371	
<code>\cs_to_str:N</code>	475, 498
<code>\cs_undefine:N</code>	2632, 2633, 2634, 2635
D	
<code>\d</code>	211
<code>\DeclareDocumentEnvironment</code>	390
dim commands:	
<code>\dim_abs:n</code>	3304, 3309
<code>\dim_add:Nn</code>	3808, 3907, 3938
<code>\dim_compare:nNnTF</code> . 919, 935, 947, 959, 1230, 1242, 1270, 1282, 1309, 1321, 1398, 1406, 1517, 1546, 3301, 3306, 3312, 3318, 3320, 3322, 3513, 3534, 3657, 3674, 3800, 3884, 3900, 3915, 3931, 4044, 4099	
<code>\dim_compare:nTF</code>	2481, 2825, 3432, 3618
<code>\dim_gset_eq:NN</code>	4053, 4108
<code>\dim_gzero:N</code>	2864, 4094, 4149
<code>\dim_new:N</code> . 67, 74, 75, 76, 94, 140, 173, 175, 176, 182	
<code>\dim_set:Nn</code> . . 486, 887, 3120, 3304, 3309, 3311, 3314, 3315, 3319, 3321, 3324, 3325, 3327, 3428, 3516, 3537, 3614, 3659, 3676, 3835, 3886, 3893, 3917, 3924, 3979, 4028, 4046, 4101, 4302	
<code>\dim_set_eq:NN</code> 582, 629, 700, 704, 3035, 3036, 3048, 3049, 3115, 3342, 3384, 3545, 3684, 3986, 3989, 3990, 4035, 4038, 4039, 4293	
<code>\dim_sub:Nn</code>	3437, 3623, 3902, 3933
<code>\dim_use:N</code> 920, 928, 1518, 1528, 2359, 2362, 2367, 3135, 3137, 3434, 3439, 3514, 3519, 3520, 3525, 3535, 3539, 3540, 3542	
<code>\dim_zero:N</code> 3376, 3548, 3685, 3809, 3810, 3811	
<code>\dim_zero_new:N</code>	455
<code>\c_zero_dim</code> 922, 936, 948, 960, 1518, 1546, 2483, 2827, 3301, 3306, 3312, 3319, 3434, 3514, 3535, 3620, 3657, 3674, 3884, 3900, 3915, 3931, 4044, 4099	
<code>\dimeval</code>	2152
E	
<code>\end</code>	2322, 2353, 3564, 3699, 4728, 4737, 4744
end internal commands:	
<code>\end__enumext_mini_page</code> . 1526, 1553, 3581, 3716, 4065, 4088, 4120, 4143	
<code>\endgroup</code>	2756
<code>\endlist</code>	378
<code>\endminipage</code>	384
<code>enumext</code>	5, <u>3406</u>
enumext internal commands:	
<code>\l__enumext__ref_the_count_tl</code>	38
<code>\l__enumext__resume_name_tl</code>	61
<code>__enumext_add_meta_key:nnn</code> . . . 123, <u>4855</u> , 4871, 4872, 4874, 4877	
<code>__enumext_add_pre_parsep:</code> . 48, 1098, <u>1100</u> , 1100	
<code>__enumext_after_args_exec:</code> 46, <u>993</u> , 1005, 3419	
<code>__enumext_after_args_exec_v:</code> <u>1009</u> , 1021, 3605	
<code>__enumext_after_args_exec_vii:</code> . . <u>1025</u> , 1049	
<code>__enumext_after_args_exec_viii:</code>	1053
<code>__enumext_after_env:nn</code> . 80, 81, 83, 98, 107, 113, <u>197</u> , 197, 2770, 3592, 4070, 4125, 4403	
<code>__enumext_after_hyperref:</code> . . . 34, 402, <u>404</u> , 404	
<code>__enumext_after_list:</code> . 97, 115, 3424, <u>3570</u> , 3570	
<code>\l__enumext_after_list_args_v_tl</code>	1023
<code>\l__enumext_after_list_args_vii_tl</code> 1051, 4361	
<code>\l__enumext_after_list_args_viii_tl</code> . . 1055, 4602	
<code>__enumext_after_list_v:</code> 3610, <u>3653</u> , 3705	

```

\__enumext_after_list_vii: 109, 4173, 4210, 4210
\__enumext_after_list_viii: .. 4425, 4464, 4464
\__enumext_after_stop_list: .. 46, 97, 993, 1001,
    3586
\__enumext_after_stop_list_v: 1009, 1017, 3723
\l__enumext_after_stop_list_v_tl ..... 1019
\__enumext_after_stop_list_vii: .. 109, 1025,
    1041, 4213
\l__enumext_after_stop_list_vii_tl ... 1043
\__enumext_after_stop_list_viii: . 1045, 4467
\l__enumext_after_stop_list_viii_tl ... 1047
\l__enumext_align_label_vii_str .. 4353, 4357
\l__enumext_align_label_viii_str . 4594, 4598
\l__enumext_align_label_X_str ..... 165
\c__enumext_all_envs_clist .. 186, 526, 792, 977,
    992, 1073, 1587
\c__enumext_all_families_seq .. 122, 4809, 4835
\l__enumext_anskey_env_bool 31, 79, 34, 283, 298,
    2686
\__enumext_anskey_env_clean_vars: . 82, 2791,
    2795, 2858
\__enumext_anskey_env_define_keys: 79, 2684,
    2693, 2764
\__enumext_anskey_env_exec: 81, 2689, 2760, 2760
\__enumext_anskey_env_make:n 65, 79, 1969, 2684,
    2684, 2692
\__enumext_anskey_env_reset_keys: 80, 81, 2729,
    2792
\__enumext_anskey_env_reset_keys:\__-
    enumext_rescan_anskey_env:n ..... 2684
\__enumext_anskey_env_save_keys: .. 81, 2772,
    2795, 2795
\__enumext_anskey_env_store: .. 82, 2788, 2795,
    2837
\__enumext_anskey_env_unknown:n 80, 2712, 2715
\__enumext_anskey_env_unknown:nn . 2717, 2719
\l__enumext_anskey_level_int .. 28, 2616, 2617
\__enumext_anskey_safe_inner: . 78, 2589, 2595,
    2614
\__enumext_anskey_safe_inner:n ..... 77
\__enumext_anskey_safe_outer: . 77, 2576, 2595,
    2595
\__enumext_anskey_show_wrap_arg:n . 76, 2508,
    2508, 2526, 2541
\__enumext_anskey_show_wrap_left:n 76, 2453,
    2522, 2522
\__enumext_anskey_unknown:n 77, 2544, 2558, 2560
\__enumext_anskey_unknown:nn . 2544, 2562, 2564
\__enumext_anskey_wrapper:n ..... 2149, 2520
\__enumext_at_begin_document:n 33, 34, 193, 193,
    375, 381
\l__enumext_base_line_fix_bool . 841, 851, 862,
    871
\__enumext_before_args_exec: .. 46, 96, 109, 993,
    993, 3511
\__enumext_before_args_exec_v: 1009, 1009, 3656
\__enumext_before_args_exec_vii: . 1025, 1025,
    4207
\__enumext_before_args_exec_viii: 1029, 4461
\__enumext_before_env:nn 79, 197, 201, 2637, 2649,
    2661, 2762
\__enumext_before_keys_exec: 46, 993, 997, 3416
\__enumext_before_keys_exec_v: 1009, 1013, 3602
\__enumext_before_keys_exec_vii ..... 1025
\__enumext_before_keys_exec_viii: . 1033, 4160
\__enumext_before_keys_exec_viii: 1037, 4412
\__enumext_before_list: ... 96, 3410, 3508, 3508
\__enumext_before_list_v: ... 3597, 3653, 3653
\__enumext_before_list_vii: ... 109, 4155, 4203,
    4203
\__enumext_before_list_viii: .. 115, 4408, 4458,
    4458
\l__enumext_before_no_starred_key_v_tl 1015
\l__enumext_before_no_starred_key_vii_-
    tl ..... 1035
\l__enumext_before_no_starred_key_viii_-
    tl ..... 1039
\l__enumext_before_starred_key_v_tl ... 1011
\l__enumext_before_starred_key_vii_tl . 1027
\l__enumext_before_starred_key_viii_tl 1031
\__enumext_calc_hspace:NNNNNNN 91, 3299, 3299,
    3330, 3335, 3377
\__enumext_check_ans_active: . 66, 96, 109, 2005,
    2005, 3512, 4206
\g__enumext_check_ans_item_tl ..... 85
\g__enumext_check_ans_key_bool 67, 68, 144, 350,
    2064, 2070, 2876
\l__enumext_check_ans_key_bool 67, 1990, 1995,
    2061, 2067
\__enumext_check_ans_key_hook: 67, 97, 109, 2058,
    2058, 3587, 4214
\__enumext_check_ans_level: 66, 2005, 2011, 2015
\__enumext_check_ans_log: 67, 68, 83, 2104, 2104,
    2880
\__enumext_check_ans_log_msg_greater: 2104,
    2110, 2123
\__enumext_check_ans_log_msg_less: 2104, 2108,
    2113
\__enumext_check_ans_log_msg_same_ok: 2104,
    2109, 2118
\__enumext_check_ans_msg_greater: 2080, 2086,
    2099
\__enumext_check_ans_msg_less: 2080, 2084, 2089
\__enumext_check_ans_msg_same_ok: 2080, 2085,
    2094
\__enumext_check_ans_show: .. 67, 82, 2080, 2080,
    2878
\l__enumext_check_answers_bool . 65, 66, 77, 87,
    144, 1967, 1994, 2009, 2296, 2320, 2327, 2351, 2578,
    2775, 2999, 3088, 3122, 4319
\__enumext_check_starred_cmd:n 32, 68, 85, 114,
    2128, 2128, 3608, 3784, 4423
\g__enumext_check_starred_cmd_int 144, 2131,
    2137, 2142, 3267, 3754, 4540
\l__enumext_check_start_line_env_tl . 32, 144,
    313, 321, 329, 2134, 2140, 2143
\l__enumext_columns_sep_v_dim 3674, 3676, 3684
\l__enumext_columns_sep_vii_dim .. 3884, 3886,
    3895, 3907, 3983, 4384
\l__enumext_columns_sep_viii_dim . 3915, 3917,
    3926, 3938, 4032, 4633
\l__enumext_columns_v_int 1363, 1381, 1549, 3672,
    3680, 3692, 3697
\l__enumext_columns_vii_int .. 3889, 3892, 3896,
    3905, 3947, 3951, 3954, 3960, 3966, 3970, 4378, 4392
\l__enumext_columns_viii_int . 3920, 3923, 3927,
    3936, 3996, 4000, 4003, 4009, 4015, 4019, 4627, 4642
\l__enumext_counter_i_tl ..... 45, 465
\l__enumext_counter_ii_tl ..... 45, 466

```


`\l__enumext_counter_iii_tl` 45, 467
`\l__enumext_counter_iv_tl` 45, 468
`\c__enumext_counter_style_tl` 30, 50, 217
`\g__enumext_counter_styles_tl` . 26, 36, 67, 476, 494
`\l__enumext_counter_v_tl` 45, 469, 720
`\l__enumext_counter_vi_tl` 45, 470
`\l__enumext_counter_vii_tl` 45, 471, 650
`\l__enumext_counter_viii_tl` 45, 472, 667
`\l__enumext_current_widest_dim` 26, 67, 500, 583, 630, 701, 705
`__enumext_def_meta_key:nnn` . . . 123, 4855, 4883, 4889, 4903
`__enumext_default_item:n` . . . 3084, 3084, 3148
`__enumext_define_counters:Nn` 26, 456, 456, 465, 466, 467, 468, 469, 470, 471, 472
`__enumext_endminipage:` . 34, 381, 384, 398, 3845, 4369, 4618
`\g__enumext_envir_name_tl` 32, 34, 284, 299, 358, 1937, 1942, 1952, 2092, 2097, 2102, 2116, 2121, 2126
`\l__enumext_envir_name_tl` . 31, 32, 34, 253, 263, 312, 320, 328, 5251, 5254, 5261, 5264, 5271, 5274, 5281, 5284, 5290, 5294, 5300, 5304, 5361, 5365
`__enumext_execute_after_env:` 33, 64, 67, 68, 78, 82, 2866, 2866, 3592, 4403
`__enumext_fake_item:` 917, 917, 3362
`\l__enumext_fake_item_indent_v_dim` 936, 941
`\l__enumext_fake_item_indent_v_tl` 938, 3250, 3254, 3262
`\l__enumext_fake_item_indent_vii_dim` 948, 953
`\l__enumext_fake_item_indent_vii_tl` 950, 4365
`\l__enumext_fake_item_indent_viii_dim` . 960, 965, 4610
`\l__enumext_fake_item_indent_viii_tl` . 962, 4608, 4613
`\l__enumext_fake_item_indent_X_tl` 98
`__enumext_fake_item_vii:` 917, 945, 3394
`__enumext_fake_item_viii:` 917, 957, 3399
`__enumext_filter_first_level:n` . . 121, 4752, 4752, 4786, 4797
`__enumext_filter_first_level_key:n` 121, 4752, 4757, 4761
`__enumext_filter_first_level_pair:nn` . 121, 4752, 4758, 4770
`__enumext_filter_save_key:n` . . 71, 2212, 2220, 2243, 2249, 2251, 2251, 4684, 4688, 4692, 4696, 4700, 4704
`__enumext_filter_save_key_key:n` . . 71, 2251, 2256, 2260
`__enumext_filter_save_key_pair:nn` 71, 2251, 2257, 2268
`__enumext_filter_series:n` 59, 1694, 1694, 1732, 1744, 1749
`__enumext_filter_series_key:n` 59, 1694, 1699, 1703
`__enumext_filter_series_pair:nn` . . 60, 1694, 1700, 1711
`\g__enumext_footnote_arg_seq` . 162, 3855, 3868, 3878
`\g__enumext_footnote_int` . 162, 3862, 3865, 3867, 3869
`\g__enumext_footnote_int_seq` . 162, 3856, 3869, 3874, 3877
`__enumext_footnotes_key_bool` 34
`\l__enumext_footnotes_key_bool` 29, 35, 112, 152, 413, 418, 427, 4332, 4373, 4582, 4622
`__enumext_footnotetext:nn` . . 3849, 3849, 3879
`__enumext_foreach_add_body:n` 124, 4904, 4964, 4967
`\l__enumext_foreach_after_tl` 4908, 4976
`\l__enumext_foreach_before_tl` 4906, 4971
`\g__enumext_foreach_default_keys_tl` 124, 124, 4926, 4947
`__enumext_foreach_keyans:nn` . 124, 4904, 4943, 4945
`\l__enumext_foreach_name_prop_tl` . 124, 4949, 4974
`\l__enumext_foreach_print_seq` 124, 4959, 4965, 4969
`\l__enumext_foreach_sep_tl` 4918, 4965
`\l__enumext_foreach_start_int` 4910, 4961
`\l__enumext_foreach_step_int` 4914, 4962
`\l__enumext_foreach_stop_int` . 4912, 4954, 4956, 4963
`__enumext_foreach_wrapper:n` 4916, 4972
`__enumext_getkeyans:nn` . . 119, 4667, 4671, 4671
`__enumext_getkeyans_aux:n` 119, 4655, 4658, 4658
`\l__enumext_hyperref_bool` . 29, 34, 35, 152, 409, 430, 447, 2498, 2987, 4314, 4569
`__enumext_hypertarget:nn` 35, 404, 432, 436, 452
`__enumext_if_is_int:n` 209
`__enumext_if_is_int:nTF` 209, 739, 753
`__enumext_internal_mini_page:` 34, 94, 109, 386, 386, 3445, 4177
`__enumext_is_not_nested:` 26, 31, 94, 109, 247, 247, 3446, 4178
`__enumext_is_on_first_level:` . 26, 31, 95, 109, 247, 273, 3452, 4190
`\g__enumext_item_anskey_int` 77, 85, 144, 345, 372, 373, 2077, 2447, 3001
`__enumext_item_answer_diff:` . . 67, 68, 82, 2073, 2073, 2873
`\g__enumext_item_answer_diff_int` . 67, 68, 144, 346, 2075, 2082, 2106
`\l__enumext_item_column_pos_vii_int` 110, 3954, 3960, 3966, 3970, 3977, 4242, 4378, 4381
`\l__enumext_item_column_pos_viii_int` . 115, 4003, 4009, 4015, 4019, 4026, 4473, 4627, 4630
`\l__enumext_item_column_pos_X_int` 165
`\g__enumext_item_count_all_vii_int` 110, 3978, 4243, 4392, 4400
`\g__enumext_item_count_all_viii_int` 115, 4027, 4474, 4641, 4650
`\g__enumext_item_count_all_X_int` 165
`\g__enumext_item_number_bool` 144
`\l__enumext_item_number_bool` 66, 150, 2027, 2032, 2036, 2040, 2053, 2621, 2675, 3091, 3125, 4322
`\g__enumext_item_number_int` 66, 67, 144, 344, 371, 373, 2026, 2031, 2035, 2039, 2052, 2077, 3090, 3124, 4321
`__enumext_item_peek_args_vii:` 110, 4244, 4246, 4246
`__enumext_item_peek_args_viii:` . 115, 4475, 4477, 4477
`__enumext_item_star_exec:` . 88, 3103, 3130, 3159
`\l__enumext_item_starred_vii_bool` 4261, 4277, 4336
`\l__enumext_item_starred_viii_bool` 4492, 4508,

4586, 4606
 \l__enumext_item_starred_X_bool 165
 __enumext_item_std:w . . 33, 87, 90, 102, 375, 379, 3094, 3100, 3128, 3250, 3254, 3262, 3821
 \g__enumext_item_symbol_aux_tl . 87, 128, 3108, 3111, 3136, 3164
 \g__enumext_item_symbol_aux_vii_tl 4285, 4338, 4341, 4345, 4347
 \g__enumext_item_symbol_aux_X_tl 165
 \l__enumext_item_symbol_sep_vii_dim . . 4294, 4302, 4344, 4346
 \l__enumext_item_symbol_vii_tl 4341
 \l__enumext_item_text_vii_box 4326, 4372
 \l__enumext_item_text_viii_box . . . 4576, 4621
 \l__enumext_item_text_X_box 165
 \l__enumext_item_width_vii_dim . . . 3893, 3902, 3981, 3989, 3990
 \l__enumext_item_width_viii_dim . . 3924, 3933, 4030, 4038, 4039
 \l__enumext_item_width_X_dim 165
 \l__enumext_itemindent_X_dim 71
 \l__enumext_itemsep_i_skip . . . 1224, 1231, 1234, 1236, 1243, 1247, 1250, 1252, 1392, 1399, 1401, 1402, 1407, 1411, 1413, 1414
 \l__enumext_itemsep_ii_skip . . 1264, 1271, 1274, 1276, 1283, 1287, 1290, 1292
 \l__enumext_itemsep_iii_skip . 1303, 1310, 1313, 1315, 1322, 1326, 1329, 1331
 \l__enumext_itemsep_vii_skip 4398
 \l__enumext_itemsep_viii_skip 4648
 \l__enumext_joined_item_aux_vii_int . . 3975, 3976, 3977, 3978, 3984
 \l__enumext_joined_item_aux_viii_int . 4024, 4025, 4026, 4027, 4033
 \l__enumext_joined_item_aux_X_int 165
 __enumext_joined_item_vii:w . . 110, 4249, 4250, 4252, 4252
 \l__enumext_joined_item_vii_int . . 3946, 3947, 3950, 3952, 3958, 3963, 3968, 3973, 3975, 3981
 __enumext_joined_item_viii:w . 115, 116, 4480, 4481, 4483, 4483
 \l__enumext_joined_item_viii_int . 3995, 3996, 3999, 4001, 4007, 4012, 4017, 4022, 4024, 4030
 \l__enumext_joined_item_X_int 165
 \l__enumext_joined_width_vii_dim . 3979, 3986, 3989, 4328, 4362
 \l__enumext_joined_width_viii_dim 4028, 4035, 4038, 4578, 4603
 \l__enumext_joined_width_X_dim 165
 __enumext_keyans_addto_prop:n 83, 2886, 2886, 3264, 3751
 __enumext_keyans_addto_seq:n . 84, 2960, 2960, 3266, 3753
 __enumext_keyans_addto_seq_link: 2960, 2981, 2983, 4539
 __enumext_keyans_anspic_code:nnn 101, 3742, 3745, 3745
 __enumext_keyans_default_item:n . . 90, 3245, 3245, 3281
 \l__enumext_keyans_env_bool 34, 3477, 3490, 3636, 3722
 __enumext_keyans_fake_item: . . 917, 933, 3352
 \l__enumext_keyans_level_h_int . . 114, 28, 660, 687, 2605, 2667, 2938, 4184, 4433, 4434
 \l__enumext_keyans_level_int . . 28, 1509, 2601, 2663, 2933, 3635, 3640, 3736
 __enumext_keyans_make_label: 37, 91, 3269, 3285, 3350
 __enumext_keyans_mini_right_cmd:n 56, 1511, 1544, 1544
 __enumext_keyans_mini_set_vskip: 53
 __enumext_keyans_minipage_add_space: 1343, 1369, 3665
 __enumext_keyans_minipage_set_skip: . 1343, 1343, 1371
 __enumext_keyans_multi_addvspace: 1142, 1153, 3689
 __enumext_keyans_multi_set_vskip: 49, 1142, 1142, 1155
 __enumext_keyans_multicols_start: 3653, 3668, 3670
 __enumext_keyans_multicols_stop: 1548, 3653, 3695, 3720
 __enumext_keyans_name_and_start: 26, 32, 114, 306, 306, 3637, 3796, 4438
 __enumext_keyans_parse_keys:n 3596, 3649, 3649
 \l__enumext_keyans_pic_above_int . 139, 3836, 3837, 3839
 \l__enumext_keyans_pic_above_skip . 102, 139, 3775, 3815
 __enumext_keyans_pic_arg_two: 102, 3773, 3803, 3803
 \l__enumext_keyans_pic_below_int . 139, 3836, 3837, 3840
 \l__enumext_keyans_pic_body_seq 101, 103, 139, 3740, 3780, 3844
 __enumext_keyans_pic_do:n 103, 3780, 3782, 3828, 3828, 3832
 \l__enumext_keyans_pic_level_int . . 28, 1493, 2609, 2671, 2889, 2928, 2963, 3051, 3791, 3792
 __enumext_keyans_pic_row:n . . . 103, 3830, 3833, 3833
 __enumext_keyans_pic_safe_exec: . 102, 3769, 3789, 3789
 __enumext_keyans_pic_skip_abs:N . 102, 3798, 3798, 3814
 \l__enumext_keyans_pic_width_dim . 139, 3835, 3842
 __enumext_keyans_pre_itemsep_skip: . . 1343, 1362, 1389
 __enumext_keyans_redefine_item: . . 91, 3269, 3269, 3349
 __enumext_keyans_ref: 41, 712, 730, 3351
 __enumext_keyans_ref:n 40, 709, 712, 712
 __enumext_keyans_safe_exec: . 3595, 3629, 3629
 __enumext_keyans_set_item_width: . 98, 3604, 3612, 3612
 __enumext_keyans_show_ans: . . 3004, 3012, 3031
 __enumext_keyans_show_item_opt: . 3004, 3019, 3262, 3765, 4609
 __enumext_keyans_show_left:n . 90, 3004, 3004, 3260, 3760
 __enumext_keyans_show_pos: . . 3004, 3016, 3044
 __enumext_keyans_starred_item:n . . 90, 3257, 3257, 3277
 __enumext_keyans_store_ref: . . 83, 2907, 2907, 3265, 3752, 4537
 __enumext_keyans_store_ref_aux_i: 84, 2907,

2919, 2922
 __enumext_keyans_store_ref_aux_ii: 84, 2907, 2948, 2950
 __enumext_keyans_unknown_keys:n 3183, 3187, 3191
 __enumext_keyans_unknown_keys:nn 3183, 3193, 3195
 __enumext_keyans_wrapper_opt:n .. 2155, 3027
 \l__enumext_label_copy_i_tl .. 2413, 2926, 2931, 2936, 2941
 \l__enumext_label_copy_v_tl 2936
 \l__enumext_label_copy_vi_tl 2931
 \l__enumext_label_copy_vii_tl 2389, 2400, 2429, 2926
 \l__enumext_label_copy_viii_tl 2941
 \l__enumext_label_copy_X_tl 154
 \l__enumext_label_fill_left_v_tl 3289
 \l__enumext_label_fill_left_X_tl 98
 \l__enumext_label_fill_right_v_tl 3296
 \l__enumext_label_fill_right_X_tl 98
 \l__enumext_label_font_style_v_tl 3290, 3764
 \l__enumext_label_font_style_vii_tl ... 4350
 \l__enumext_label_font_style_viii_tl .. 4591
 \l__enumext_label_i_tl 575
 \l__enumext_label_ii_tl 575
 \l__enumext_label_iii_tl 575
 \l__enumext_label_iv_tl 575
 __enumext_label_style:Nnn 26, 36, 489, 489, 504, 580, 627, 698, 702
 \l__enumext_label_v_tl .. 83, 84, 695, 2894, 2968, 3038, 3078, 3259, 3263, 3599, 3759, 3761
 \l__enumext_label_vi_tl . 83, 84, 695, 2891, 2965, 3759, 3761, 3765
 \l__enumext_label_vii_tl . 622, 4272, 4297, 4304
 \l__enumext_label_viii_tl 622, 4503, 4531, 4535
 \l__enumext_label_width_by_box .. 67, 485, 486
 __enumext_label_width_by_box:Nn 36, 483, 483, 488, 500, 763
 \l__enumext_labelsep_i_dim ... 3036, 3041, 3049, 3081, 4543, 4558
 \l__enumext_labelsep_v_dim 3679
 \l__enumext_labelsep_vii_dim . 2513, 3036, 3049, 3888, 3898, 3982, 4295, 4330, 4360
 \l__enumext_labelsep_viii_dim 3919, 3929, 4031, 4580, 4601, 4610
 \l__enumext_labelwidth_i_dim . 3035, 3041, 3048, 3081, 4543, 4558
 \l__enumext_labelwidth_v_dim 3679
 \l__enumext_labelwidth_vii_dim ... 2513, 3035, 3048, 3888, 3897, 3982, 4329, 4353, 4357
 \l__enumext_labelwidth_viii_dim .. 3919, 3928, 4031, 4579, 4594, 4598
 \l__enumext_leftmargin_tmp_v_bool . 102, 3805
 \l__enumext_leftmargin_tmp_X_bool 71
 \l__enumext_leftmargin_tmp_X_dim 71
 \l__enumext_leftmargin_X_dim 71
 __enumext_level: 205, 205, 604, 607, 608, 617, 619, 920, 924, 928, 995, 999, 1003, 1007, 1090, 1092, 1094, 1096, 1129, 1131, 1133, 1135, 1140, 1175, 1181, 1186, 1188, 1191, 1194, 1207, 1210, 1518, 1522, 1528, 1591, 1593, 1595, 1598, 1605, 1607, 1609, 1612, 2207, 2209, 2211, 2239, 2240, 2242, 2298, 2306, 2310, 2314, 2517, 2518, 3093, 3094, 3098, 3099, 3100, 3108, 3116, 3117, 3120, 3127, 3128, 3132, 3135, 3137, 3155, 3156, 3157, 3160, 3163, 3413, 3415, 3434, 3439, 3483, 3496, 3503, 3514, 3516, 3519, 3520, 3522, 3525, 3532, 3535, 3537, 3539, 3540, 3541, 3542, 3545, 3550, 3556, 3562, 3567, 3572
 \l__enumext_level_h_int 109, 28, 256, 279, 293, 643, 680, 1500, 2023, 2043, 2408, 2641, 2653, 3491, 4179, 4180
 \l__enumext_level_int . 94, 28, 207, 266, 278, 294, 388, 1102, 1220, 1499, 2017, 2049, 2385, 2395, 2401, 2407, 2414, 2423, 2428, 2640, 2652, 2868, 3365, 3447, 3448, 3459, 3467, 3481, 3494, 3546, 3644, 3732, 4223, 4233, 4446, 5291, 5295, 5301, 5305
 __enumext_list_arg_two_i: 3331
 __enumext_list_arg_two_ii: 3331
 __enumext_list_arg_two_iii: 3331
 __enumext_list_arg_two_iv: 3331
 __enumext_list_arg_two_v: . 91, 3331, 3601, 3806
 __enumext_list_arg_two_vii: 3371, 4159
 __enumext_list_arg_two_viii: 3371, 4411
 \l__enumext_listoffset_v_dim . 3620, 3625, 3681
 \l__enumext_listparindent_vii_dim 4363
 \l__enumext_listparindent_viii_dim ... 4604
 __enumext_log_answer_vars: . 33, 360, 368, 2875
 __enumext_log_global_vars: . 33, 360, 360, 2874
 __enumext_make_label 3140
 __enumext_make_label: 37, 88, 3151, 3360
 \l__enumext_mark_answer_sym_tl 73, 2161, 2364, 2530, 3053, 3066, 4547
 \l__enumext_mark_position_str 128, 2165, 2166, 2192, 2193, 2362
 \l__enumext_mark_ref_sym_tl .. 2178, 2503, 2995
 \l__enumext_meta_path_tl . 124, 4879, 4880, 4882, 4883
 \c__enumext_meta_paths_prop 123, 4855
 __enumext_mini_addvspace_vii: 55, 1479, 1479, 4056
 __enumext_mini_addvspace_viii: 55, 1479, 1485, 4111
 __enumext_mini_env* 386
 __enumext_mini_page 1528, 1555, 3525, 3666, 4058, 4074, 4113, 4129
 __enumext_mini_right_cmd:n 56, 1513, 1515, 1515
 __enumext_mini_set_vskip_vii: 54, 1422, 1422, 1481
 __enumext_mini_set_vskip_viii: 54, 1422, 1444, 1487
 __enumext_minipage:w 34, 381, 383, 392, 3842, 4362, 4603
 \l__enumext_minipage_active_v_bool 3663, 3686, 3707
 \g__enumext_minipage_active_vii_bool .. 107, 4067, 4072, 4091
 \l__enumext_minipage_active_vii_bool . 4052, 4063
 \g__enumext_minipage_active_viii_bool 4122, 4127, 4146
 \l__enumext_minipage_active_viii_bool 4107, 4118
 \g__enumext_minipage_active_X_bool ... 165
 \l__enumext_minipage_active_X_bool 86
 __enumext_minipage_add_space: .. 50, 96, 1171, 1197, 3524
 \g__enumext_minipage_after_skip 86, 1426, 1438, 4089, 4144

`\l__enumext_minipage_after_skip` . . 50, 97, 86, 1184, 1224, 1226, 1231, 1234, 1238, 1243, 1247, 1250, 1254, 1266, 1271, 1274, 1278, 1283, 1287, 1290, 1294, 1305, 1310, 1313, 1317, 1322, 1326, 1329, 1333, 1345, 1359, 1392, 1394, 1399, 1401, 1403, 1407, 1411, 1413, 1415, 1446, 1459, 1473, 1524, 1551, 3717
`\g__enumext_minipage_center_vii_bool` . 4076, 4092
`\g__enumext_minipage_center_viii_bool` 4131, 4147
`\g__enumext_minipage_center_X_bool` . . . 165
`\l__enumext_minipage_hsep_v_dim` 3661
`\l__enumext_minipage_hsep_vii_dim` 4050
`\l__enumext_minipage_hsep_viii_dim` . . . 4105
`\l__enumext_minipage_left_skip` 86, 1346, 1424, 1429, 1433, 1447, 1451, 1465, 1483, 1489
`\l__enumext_minipage_left_v_dim` . . 3659, 3666
`\l__enumext_minipage_left_vii_dim` 4046, 4058
`\l__enumext_minipage_left_viii_dim` 4101, 4113
`\l__enumext_minipage_left_X_dim` 86
`\g__enumext_minipage_right_skip` 86, 1425, 1430, 1434, 4075, 4130
`\l__enumext_minipage_right_skip` . 50, 86, 1173, 1179, 1184, 1186, 1188, 1347, 1348, 1354, 1359, 1360, 1361, 1366, 1448, 1455, 1469, 1530, 1557
`\l__enumext_minipage_right_v_dim` . 1546, 1555, 3657, 3661
`\g__enumext_minipage_right_vii_dim` 106, 4054, 4074, 4094
`\l__enumext_minipage_right_vii_dim` 106, 4044, 4049, 4055
`\g__enumext_minipage_right_viii_dim` . . 4109, 4129, 4149
`\l__enumext_minipage_right_viii_dim` . . 4099, 4104, 4110
`\g__enumext_minipage_right_X_dim` 165
`\g__enumext_minipage_right_X_skip` 165
`__enumext_minipage_set_skip:` . 50, 1171, 1171, 1199
`\g__enumext_minipage_stat_int` 96, 86, 1535, 1562, 3523, 3574, 3579, 3664, 3709, 3714
`\l__enumext_miniright_code_vii_box` 4083, 4087
`\g__enumext_miniright_code_vii_tl` 107, 4078, 4085, 4093
`\l__enumext_miniright_code_viii_box` . . 4138, 4142
`\g__enumext_miniright_code_viii_tl` 4133, 4140, 4148
`\l__enumext_miniright_code_X_box` 165
`__enumext_multi_addvspace:` . 49, 97, 1124, 1124, 3553
`__enumext_multi_set_vskip:` 48, 1088, 1088, 1126
`\l__enumext_multicols_above_ii_skip` . . 1107
`\l__enumext_multicols_above_iii_skip` . . 1113
`\l__enumext_multicols_above_iv_skip` . . 1119
`\l__enumext_multicols_above_v_skip` 1144, 1158, 1169, 1360
`\l__enumext_multicols_above_X_skip` 79
`\l__enumext_multicols_below_ii_skip` . . 1227, 1236, 1240, 1252, 1257
`\l__enumext_multicols_below_iii_skip` . 1267, 1276, 1280, 1292, 1297
`\l__enumext_multicols_below_iv_skip` . . 1306, 1315, 1319, 1331, 1336
`\l__enumext_multicols_below_v_skip` 1148, 1162, 1361, 1395, 1402, 1404, 1414, 1417, 3702
`\l__enumext_multicols_below_X_skip` 79
`\g__enumext_multicols_right_X_skip` 79
`__enumext_multicols_start:` 96, 3527, 3529, 3529
`__enumext_multicols_stop:` 97, 1520, 3559, 3559, 3584
`__enumext_nested_base_line_fix:` . 43, 95, 109, 837, 847, 3463, 4200
`__enumext_newlabel:nn` 29, 35, 74, 440, 440, 2439, 2954
`\l__enumext_newlabel_arg_one_tl` 29, 35, 74, 84, 154, 2432, 2440, 2502, 2943, 2955, 2993
`\l__enumext_newlabel_arg_two_tl` 29, 35, 73, 154, 2388, 2398, 2411, 2426, 2441, 2930, 2935, 2940, 2956
`__enumext_parse_foreach_keys:n` . . 4904, 4920, 4937
`__enumext_parse_foreach_keys:nn` . 4904, 4927, 4939
`__enumext_parse_keys:n` 43, 60, 3409, 3454, 3454
`__enumext_parse_keys_vii:n` . 43, 60, 4154, 4192, 4192
`__enumext_parse_keys_viii:n` . 4407, 4451, 4451
`__enumext_parse_save_key:n` 70, 2232, 2237, 2237
`__enumext_parse_save_key_vii:n` 70, 2227, 2237, 2245
`__enumext_parse_series:n` 60, 95, 109, 1720, 1720, 3462, 4198
`__enumext_parse_store_keys:n` 95
`\l__enumext_parsep_i_skip` 1105, 1107
`\l__enumext_parsep_ii_skip` 1111, 1113
`\l__enumext_parsep_iii_skip` 1117, 1119
`\l__enumext_parsep_vii_skip` 4364
`\l__enumext_parsep_viii_skip` 4605
`\l__enumext_partopsep_v_skip` . 1160, 1164, 1356, 1379
`\l__enumext_partopsep_viii_skip` 1457
`__enumext_phantomsection:` 35, 404, 433, 437, 453
`__enumext_pre_itemsep_skip:` 50, 51, 1189, 1218, 1218
`__enumext_print_footnote:` . . 3849, 3872, 4375, 4624
`__enumext_print_keyans_box:NN` 73, 2356, 2356, 2369, 2513, 2516, 3040, 3080, 4543, 4558
`\l__enumext_print_keyans_i_tl` 4689, 4711
`\l__enumext_print_keyans_ii_tl` . . 4693, 4712
`\l__enumext_print_keyans_iii_tl` . . 4697, 4713
`\l__enumext_print_keyans_iv_tl` . . 4701, 4714
`\l__enumext_print_keyans_starred_tl` 119, 120, 128, 4685, 4733
`\l__enumext_print_keyans_vii_tl` 119, 4705, 4715
`\l__enumext_print_keyans_X_tl` 128
`__enumext_printkeyans:nnn` 120, 4716, 4719, 4719
`__enumext_redefine_item:` . 88, 3140, 3140, 3359
`\l__enumext_ref_key_arg_tl` 38, 50, 220, 597, 598, 611, 642, 645, 656, 662, 673, 714, 715, 726
`\l__enumext_ref_the_count_tl` . 38, 50, 604, 607, 610, 650, 652, 655, 667, 669, 672, 720, 722, 725
`__enumext_regex_counter_style:` . . 30, 38, 215, 215, 605, 651, 668, 721
`__enumext_register_counter_style:Nn` . . 473, 473, 478, 479, 480, 481, 482
`__enumext_remove_extra_parsep_vii:` . . 4170, 4387, 4387

__enumext_remove_extra_parsep_viii: . 4422, 4636, 4636
 __enumext_renew_footnote: . . . 3849, 3853, 4334, 4584
 \l__enumext_renew_the_count_v_tl 723, 732, 734
 \l__enumext_renew_the_count_vii_tl 653, 682, 684
 \l__enumext_renew_the_count_viii_tl 670, 689, 691
 \l__enumext_renew_the_count_X_tl 50
 __enumext_rescan_anskey_env:n . . 80, 82, 2750, 2845, 2853
 __enumext_reset_global_bool: . . 336, 339, 348
 __enumext_reset_global_int: . . . 336, 338, 342
 __enumext_reset_global_tl: . . . 336, 340, 354
 __enumext_reset_global_vars: . 33, 83, 336, 336, 2883
 \l__enumext_resume_active_bool 60, 62, 61, 1724, 1844
 __enumext_resume_counter: . 62, 1842, 1848, 1855
 __enumext_resume_counter:n . 60, 62, 1813, 1818, 1842, 1842, 1912, 1920
 __enumext_resume_counter_save_ans: . . 62, 63, 1842, 1853, 1885
 __enumext_resume_counter_series: 62, 63, 1842, 1851, 1868
 \g__enumext_resume_int . . . 61, 1765, 1859, 1860
 __enumext_resume_last:n . . 60, 1720, 1726, 1739
 \l__enumext_resume_name_tl 61, 1761, 1769, 1772, 1788, 1796, 1799, 1845, 1846, 1874, 1881
 __enumext_resume_save_counter: . . 61, 97, 109, 1752, 1752, 3590, 4217
 __enumext_resume_series:n . 62, 1688, 1809, 1809
 __enumext_resume_starred: . 63, 1689, 1906, 1906
 \g__enumext_resume_vii_int 61, 1792, 1864, 1865
 \l__enumext_rightmargin_vii_dim . . 3900, 3904, 3909
 \l__enumext_rightmargin_viii_dim . 3931, 3935, 3940
 __enumext_safe_exec: . . 34, 94, 3408, 3443, 3443
 __enumext_safe_exec_vii: . 34, 4153, 4175, 4175
 __enumext_safe_exec_viii: 114, 4406, 4427, 4427
 \l__enumext_series_name_tl 62
 \l__enumext_series_str . . 61, 95, 109, 1686, 1722, 1730, 1731, 1733, 1735, 1756, 1759, 1763, 1783, 1786, 1790, 3458, 4196
 __enumext_set_error:nn 4841, 4851, 4853
 __enumext_set_item_width: . 94, 3418, 3426, 3426
 __enumext_set_parse:n 4825, 4841, 4841
 \l__enumext_setkey_tmpa_int . . . 119, 4818, 4822
 \l__enumext_setkey_tmpa_seq . . 119, 4816, 4826, 4832, 4834, 4836, 4848
 \l__enumext_setkey_tmpa_tl . . . 119, 4824, 4828
 \l__enumext_setkey_tmpb_seq . . 119, 4817, 4820, 4824, 4825
 \l__enumext_setkey_tmpb_tl 119, 4843, 4845, 4846
 \l__enumext_show_answer_bool . 2172, 2196, 2524, 3010, 3024, 3756, 4541
 __enumext_show_length:nnn . . 45, 223, 223, 5062, 5063, 5064, 5065, 5066, 5067, 5068, 5069, 5070, 5071, 5077, 5078, 5079, 5080, 5081, 5082, 5083, 5084, 5085, 5086
 \l__enumext_show_position_bool . . 2175, 2199, 2528, 3014, 3025, 3757, 4545
 \g__enumext_standar_bool 31, 94, 34, 255, 258, 277, 351, 1754, 1819, 1831, 1857, 1870, 1908, 2048, 2062, 2393, 2406, 2421, 3478
 \l__enumext_standar_bool . 94, 97, 34, 2394, 3450, 3589, 4189
 \l__enumext_standar_first_bool 31, 95, 34, 282, 850, 1741, 1888, 1950, 1957
 __enumext_standar_item_vii:w . 110, 111, 4257, 4259, 4259
 __enumext_standar_item_viii:w 116, 4488, 4490, 4490
 __enumext_standar_ref: 39, 595, 615, 3361
 __enumext_standar_ref:n 38, 587, 595, 595
 \g__enumext_standar_series_tl . 61, 1743, 1744, 1910, 1913
 __enumext_standar_unknown_keys:n 3223, 3227, 3231
 __enumext_standar_unknown_keys:nn 3223, 3233, 3235
 \g__enumext_starred_bool 31, 109, 34, 265, 268, 292, 352, 1781, 1824, 1835, 1862, 1877, 1916, 2022, 2068, 2384, 2924, 4095
 \l__enumext_starred_bool 109, 114, 34, 1505, 2422, 2457, 2463, 2511, 2799, 2804, 3033, 3046, 3451, 4188, 4216, 4439, 4443
 __enumext_starred_columns_set_vii: . . 3882, 3882, 4162
 __enumext_starred_columns_set_viii: . 3882, 3913, 4414
 \l__enumext_starred_first_bool 31, 109, 34, 297, 861, 1746, 1897, 1950, 1957
 __enumext_starred_item:nn . . . 3103, 3103, 3146
 __enumext_starred_item_exec: . 117, 4533, 4533, 4588
 __enumext_starred_item_vii:w . 110, 111, 4256, 4275, 4275
 __enumext_starred_item_vii_aux_i:w . . 4275, 4280, 4283
 __enumext_starred_item_vii_aux_ii:w . 4275, 4281, 4286, 4288
 __enumext_starred_item_vii_aux_iii:w 4275, 4291, 4300
 __enumext_starred_item_viii:w 116, 4487, 4506, 4506
 __enumext_starred_item_viii_aux_i:w . . 116, 4506, 4511, 4514
 __enumext_starred_item_viii_aux_ii:w . 116, 4506, 4512, 4526, 4528
 __enumext_starred_joined_item_vii:n 105, 110, 3944, 3944, 4254
 __enumext_starred_joined_item_viii:n . 105, 116, 3944, 3993, 4485
 __enumext_starred_ref: 40, 640, 678, 3391
 __enumext_starred_ref:n 39, 634, 640, 640
 \g__enumext_starred_series_tl . 61, 1748, 1749, 1918, 1921
 __enumext_starred_unknown_keys:n 3205, 3207, 3209
 __enumext_starred_unknown_keys:nn 3205, 3211, 3213
 __enumext_start_from:NNn 41, 737, 737, 750, 772, 778
 \l__enumext_start_i_int 1860, 1872, 1891
 __enumext_start_item_tmp_vii: 108, 4165, 4239,

4239
 __enumext_start_item_tmp_viii: .. 114, 4417, 4470, 4470
 __enumext_start_item_vii:w 111, 112, 4267, 4272, 4297, 4304, 4306, 4306
 __enumext_start_item_viii:w .. 116, 4498, 4503, 4531, 4561, 4561
 \g__enumext_start_line_tl 31, 34, 285, 300, 357, 2092, 2097, 2102, 2116, 2121, 2126
 __enumext_start_list:nn .. 33, 91, 102, 375, 377, 3412, 3598, 3770, 4157, 4409
 __enumext_start_mini_vii: 109, 4042, 4042, 4208
 __enumext_start_mini_viii: ... 115, 4097, 4097, 4462
 __enumext_start_save_ans_msg: 64, 1934, 1934, 1959
 __enumext_start_store_level: . 95, 3411, 3472, 3472
 __enumext_start_store_level_vii: 110, 4156, 4219, 4219
 \l__enumext_start_vii_int ... 1865, 1879, 1900
 \l__enumext_start_X_int 98
 __enumext_stop_item_tmp_vii: .. 108, 110, 112, 4164, 4169, 4241, 4308
 __enumext_stop_item_tmp_viii: 114, 115, 4416, 4421, 4472, 4563
 __enumext_stop_item_vii: 112, 113, 4308, 4367, 4367
 __enumext_stop_item_viii: 118, 4563, 4616, 4616
 __enumext_stop_list: .. 33, 375, 378, 3422, 3609, 3783, 4171, 4424
 __enumext_stop_mini_vii: 107, 109, 4042, 4061, 4212
 __enumext_stop_mini_viii: 115, 4097, 4116, 4466
 __enumext_stop_save_ans_msg: . 64, 1934, 1939, 2872
 __enumext_stop_store_level: .. 95, 3423, 3472, 3501
 __enumext_stop_store_level_vii: . 110, 4172, 4219, 4229
 \l__enumext_store_active_bool 28, 65, 110, 1889, 1898, 1966, 2597, 3476, 3489, 3631, 3639, 3728, 3787, 4221, 4231, 4429, 4445
 __enumext_store_active_keys:n .. 70, 95, 2205, 2205, 3469
 __enumext_store_active_keys_vii:n . 70, 109, 2205, 2215, 4199
 __enumext_store_addto_prop:n 71, 83, 2280, 2280, 2288, 2448, 2905, 4536
 __enumext_store_addto_seq:n 72, 85, 2289, 2289, 2293, 2300, 2314, 2322, 2331, 2345, 2353, 2506, 2998
 \l__enumext_store_anskey_arg_tl .. 28, 75, 110, 2454, 2459, 2461, 2466, 2473, 2476, 2486, 2491, 2494, 2500, 2506
 __enumext_store_anskey_code:n 74, 77, 82, 2445, 2445, 2590, 2843, 2851
 \l__enumext_store_anskey_env_tl .. 28, 81, 110, 2773, 2777, 2783, 2845, 2853
 \l__enumext_store_anskey_opt_tl 28, 81, 82, 110, 2774, 2801, 2807, 2814, 2820, 2830, 2840, 2849
 __enumext_store_anskey_safe_outer: 77
 \g__enumext_store_columns_break_bool . 2697, 2798, 2860
 \l__enumext_store_columns_break_bool . 2456, 2546
 \l__enumext_store_current_label_tl 28, 83-85, 116, 110, 2888, 2891, 2894, 2901, 2903, 2905, 2962, 2965, 2968, 2974, 2979, 2989, 2998, 4516, 4521, 4522, 4535, 4536, 4538
 \l__enumext_store_current_label_tmp_tl . 28, 110, 3259, 3263
 \l__enumext_store_current_opt_arg_tl 28, 116, 110, 3008, 3021, 3027, 4524
 __enumext_store_internal_ref: .. 73, 74, 2370, 2370, 2451
 \g__enumext_store_item_join_int .. 2700, 2805, 2809, 2861
 \l__enumext_store_item_join_int .. 2464, 2468, 2549
 \g__enumext_store_item_star_bool . 2702, 2812, 2862
 \l__enumext_store_item_star_bool . 2471, 2551
 \g__enumext_store_item_symbol_sep_dim 2707, 2827, 2832, 2864
 \l__enumext_store_item_symbol_sep_dim 2483, 2488, 2556
 \g__enumext_store_item_symbol_tl . 2705, 2818, 2822, 2863
 \l__enumext_store_item_symbol_tl . 2474, 2478, 2554
 \l__enumext_store_keyans_item_opt_sep_tl 2158, 2899, 2901, 2972, 2976, 4519, 4521
 __enumext_store_level_close: . 72, 2294, 2318, 3505
 __enumext_store_level_close_vii: . 72, 2325, 2349, 4235
 __enumext_store_level_open: 72, 95, 2294, 2294, 3484, 3497
 __enumext_store_level_open_vii: .. 72, 2325, 2325, 4225
 \g__enumext_store_name_tl 28, 65, 110, 356, 363, 364, 365, 366, 1942, 1968, 2091, 2096, 2101, 2115, 2120, 2125, 2870
 \l__enumext_store_name_tl 28, 64, 66, 110, 1775, 1778, 1802, 1805, 1893, 1902, 1937, 1946, 1947, 1968, 1969, 1970, 1972, 1973, 1975, 1977, 1978, 1980, 1982, 1983, 2007, 2282, 2284, 2291, 2434, 2435, 2536, 2779, 2945, 2946, 3059, 3072, 4553
 \l__enumext_store_ref_key_bool 74, 2181, 2449, 2497, 2909, 2986
 \l__enumext_store_save_key_vii_bool .. 2217, 2247
 \l__enumext_store_save_key_vii_tl 2219, 2220, 2248, 2249, 2329, 2337, 2341, 2345
 \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, 1944, 1960, 1964
 __enumext_storing_set:n .. 64, 1929, 1944, 1944
 \l__enumext_the_counter_v_tl 722
 \l__enumext_the_counter_vii_tl 652
 \l__enumext_the_counter_viii_tl 669
 \l__enumext_the_counter_X_tl 50
 __enumext_tmp:n 45, 49, 54, 60, 71, 78, 79, 85, 92, 97, 98, 109, 131, 138, 157, 161, 165, 185, 837, 846, 1682, 1693, 1925, 1933, 1986, 2004, 2145, 2186, 2187, 2204, 2223, 2236, 2372, 2379, 2380, 2401, 2414, 2417, 2428, 2911, 2918, 3183, 3190, 3223, 3230, 3331, 3370, 3371, 3405

<code>__enumext_tmp:nn</code>	505, 526, 527, 558, 559, 574, 767, 792, 873, 895, 896, 916, 969, 977, 978, 992, 1057, 1073, 1074, 1087, 1571, 1587, 3167, 3182
<code>__enumext_tmp:nnn</code>	575, 591, 592, 593, 594, 622, 638, 639
<code>__enumext_tmp:nnnnn</code>	793, 818, 821, 824, 826, 828, 831, 834
<code>__enumext_tmp:w</code>	4664, 4667
<code>\l__enumext_tmpa_vii_int</code>	3892, 3895, 3904, 3935
<code>\l__enumext_tmpa_viii_int</code>	3923, 3926
<code>\l__enumext_tmpa_X_dim</code>	165
<code>\l__enumext_tmpa_X_int</code>	165
<code>\l__enumext_topsep_v_skip</code>	1146, 1150, 1350, 3786, 3818
<code>\l__enumext_topsep_vii_skip</code>	1427, 1436, 1440
<code>\l__enumext_topsep_viii_skip</code>	1449, 1471, 1475
<code>__enumext_undefine_anskey_env:</code>	78, 83, 2630, 2630, 2881
<code>__enumext_unskip_unkern:</code>	31, 229, 229, 1138, 1167, 1200, 1372, 3565, 3566, 3580, 3700, 3701, 3715
<code>\l__enumext_vspace_a_star_v_bool</code>	1620
<code>\l__enumext_vspace_a_star_vii_bool</code>	1642
<code>\l__enumext_vspace_a_star_viii_bool</code>	1653
<code>\l__enumext_vspace_a_star_X_bool</code>	98
<code>__enumext_vspace_above:</code>	57, 96, 1588, 1588, 3510
<code>__enumext_vspace_above_v:</code>	58, 1616, 1616, 3655
<code>\l__enumext_vspace_above_v_skip</code>	1618, 1622, 1624
<code>__enumext_vspace_above_vii:</code>	58, 109, 1638, 1638, 4205
<code>\l__enumext_vspace_above_vii_skip</code>	1640, 1644, 1646
<code>__enumext_vspace_above_viii:</code>	58, 1638, 1649, 4460
<code>\l__enumext_vspace_above_viii_skip</code>	1651, 1655, 1657
<code>\l__enumext_vspace_b_star_v_bool</code>	1631
<code>\l__enumext_vspace_b_star_vii_bool</code>	1664
<code>\l__enumext_vspace_b_star_viii_bool</code>	1675
<code>\l__enumext_vspace_b_star_X_bool</code>	98
<code>__enumext_vspace_below:</code>	57, 97, 1602, 1602, 3588
<code>__enumext_vspace_below_v:</code>	58, 1627, 1627, 3724
<code>\l__enumext_vspace_below_v_skip</code>	1629, 1633, 1635
<code>__enumext_vspace_below_vii:</code>	59, 109, 1660, 1660, 4215
<code>\l__enumext_vspace_below_vii_skip</code>	1662, 1666, 1668
<code>__enumext_vspace_below_viii:</code>	59, 1660, 1671, 4468
<code>\l__enumext_vspace_below_viii_skip</code>	1673, 1677, 1679
<code>__enumext_widest_from:nnNn</code>	41, 751, 751, 766, 785
<code>\g__enumext_widest_label_tl</code>	26, 36, 67, 493, 497, 501
<code>\l__enumext_wrap_label_opt_v_bool</code>	3253
<code>\l__enumext_wrap_label_opt_vii_bool</code>	111, 4266
<code>\l__enumext_wrap_label_opt_viii_bool</code>	116, 4497
<code>\l__enumext_wrap_label_opt_X_bool</code>	98
<code>\l__enumext_wrap_label_v_bool</code>	3249, 3253, 3261, 3291
<code>\l__enumext_wrap_label_vii_bool</code>	111, 4265, 4270, 4278, 4351
<code>\l__enumext_wrap_label_viii_bool</code>	116, 4496, 4501, 4509, 4592
<code>\l__enumext_wrap_label_X_bool</code>	98
<code>__enumext_wrapper_label_v:n</code>	3293, 3765
<code>__enumext_wrapper_label_vii:n</code>	4354
<code>__enumext_wrapper_label_viii:n</code>	4595
<code>\l__enumext_write_aux_file_tl</code>	29, 74, 84, 154, 2437, 2443, 2952, 2958
<code>enumext*</code>	5, 4151
<code>enumXi</code>	465
<code>enumXii</code>	465
<code>enumXiii</code>	465
<code>enumXiv</code>	465
<code>enumXv</code>	465
<code>enumXvi</code>	465
<code>enumXvii</code>	465
<code>enumXviii</code>	465
Environments provide by <code>enumext</code> :	
<code>anskey*</code>	28, 65, 73, 74, 76, 78, 79, 81, 83, 95, 110, 120, 125, 128
<code>enumext*</code>	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, 103–105, 107, 110, 112–115, 117, 119–121, 123, 126, 129, 131
<code>enumext</code>	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, 104, 106, 109, 110, 114, 119, 121, 123, 126, 128, 130
<code>keyans*</code>	25, 26, 28–32, 36, 39–42, 44, 45, 47, 54, 55, 58, 59, 65, 68, 69, 71, 79, 83, 89, 93, 103–105, 114, 126, 129, 131
<code>keyanspic</code>	25, 26, 28, 29, 32, 36, 37, 40, 65, 68, 71, 72, 79, 83–85, 100–102, 129
<code>keyans</code>	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, 98–102, 106, 115, 126, 129
Environments:	
<code>list</code>	30, 33, 91, 94
<code>lrbox</code>	112
<code>minipage</code>	30, 34, 47, 50, 51, 100–104, 112, 113, 118
<code>multicols</code>	48–51, 56, 96, 97
<code>scontents</code>	79, 81
exp commands:	
<code>\exp_after:wN</code>	4667
<code>\exp_args:Ne</code>	2842, 2850, 3466, 4655
<code>\exp_args:Nv</code>	2562, 2717, 3193, 3211, 3233, 4939
<code>\exp_not:N</code>	58, 496, 610, 655, 672, 725, 926, 940, 941, 952, 953, 964, 965, 2502, 2533, 2534, 2991, 3056, 3057, 3069, 3070, 4550, 4551, 4664
<code>\exp_not:n</code>	287, 302, 315, 323, 331, 549, 569, 610, 611, 655, 656, 672, 673, 725, 726, 927, 1709, 1718, 2169, 2266, 2278, 2440, 2468, 2478, 2488, 2502, 2503, 2809, 2822, 2832, 2955, 2993, 2995, 4768, 4778, 4971, 4976
F	
<code>\fbox</code>	2152
<code>\fboxrule</code>	2152
<code>\fboxsep</code>	2152
file commands:	
<code>\file_input_stop:</code>	5375
<code>first</code>	978
<code>font</code>	505
<code>\footnote</code>	103
<code>\footnote</code>	103, 3857
<code>\footnotemark</code>	3867
<code>\footnotesize</code>	2534, 3057, 3070, 4551
<code>\footnotetext</code>	3851
<code>\foreachkeyans</code>	16, 124, 4904

G

\getkeyans 16, 119, [4653](#)
 group commands:
 \group_begin: .. 2532, 2577, 2752, 2839, 3055, 3068,
 [4349](#), [4549](#), [4590](#), [4710](#)
 \group_end: 2539, 2593, 2856, 3062, 3075, 4359, 4556,
 [4600](#), [4717](#)

H

\hbadness [4371](#), [4620](#)
 hbox commands:
 \hbox_set:Nn [485](#)
 \hbox_set_end: [4370](#), [4619](#)
 \hbox_set_to_wd:Nnw [4326](#), [4576](#)
 \hfill [535](#), [539](#), [544](#), [545](#), [1527](#), [1554](#), [2502](#), [2991](#), [4066](#), [4121](#)
 hook commands:
 \hook_gput_code:nnn 9, 195, 199, 203, [402](#)
 \hook_gremove_code:nn [81](#), [2768](#)
 \hook_gset_rule:nnnn [403](#)
 \hook_if_empty:nTF [2766](#)
 \hyperlink [75](#), [85](#)
 \hyperlink [2502](#), [2991](#)
 \hypertarget [35](#)
 \hypertarget [432](#)

I

\IfHyperBoolean [410](#)
 \IfPackageLoadedTF [11](#), [19](#), [406](#), [420](#)
 \ignorespaces [929](#), [4166](#), [4418](#)
 \inputlineno [287](#), [302](#), [315](#), [323](#), [331](#)
 int commands:
 \int_add:Nn [3977](#), [4026](#)
 \int_case:nn ... [1102](#), [1220](#), [2017](#), [2043](#), [2082](#), [2106](#)
 \int_case:nnTF [231](#)
 \int_compare:nNnTF .. [388](#), [643](#), [660](#), [680](#), [687](#), [1190](#),
 [1209](#), [1363](#), [1381](#), [1493](#), [1509](#), [1521](#), [1549](#), [2130](#), [2136](#),
 [2601](#), [2605](#), [2609](#), [2617](#), [2663](#), [2667](#), [2671](#), [2868](#), [2889](#),
 [2928](#), [2933](#), [2938](#), [2963](#), [3051](#), [3448](#), [3459](#), [3481](#), [3494](#),
 [3531](#), [3546](#), [3561](#), [3574](#), [3640](#), [3644](#), [3672](#), [3697](#), [3709](#),
 [3732](#), [3736](#), [3792](#), [3947](#), [3957](#), [3973](#), [3996](#), [4006](#), [4022](#),
 [4180](#), [4184](#), [4223](#), [4233](#), [4377](#), [4389](#), [4434](#), [4446](#), [4626](#),
 [4638](#), [4822](#), [4954](#)
 \int_compare_p:nNn ... [256](#), [266](#), [278](#), [279](#), [293](#), [294](#),
 [1499](#), [1500](#), [2023](#), [2049](#), [2385](#), [2395](#), [2407](#), [2408](#), [2423](#),
 [2464](#), [2640](#), [2641](#), [2652](#), [2653](#), [2805](#), [3491](#)
 \int_decr:N [3976](#), [4025](#)
 \int_eval:n .. [373](#), [780](#), [2284](#), [2435](#), [2534](#), [2946](#), [3057](#),
 [3070](#), [3346](#), [3390](#), [3965](#), [4014](#), [4551](#)
 \int_from_alph:n [745](#), [759](#)
 \int_from_roman:n [747](#), [761](#)
 \int_gadd:Nn [3978](#), [4027](#)
 \int_gdecr:N [2026](#), [2031](#), [2035](#), [2039](#), [2052](#)
 \int_gincr:N [1859](#), [1864](#), [2447](#), [3001](#), [3090](#), [3124](#), [3267](#),
 [3523](#), [3664](#), [3754](#), [4243](#), [4321](#), [4474](#), [4540](#)
 \int_gset:Nn [2075](#), [3865](#)
 \int_gset_eq:NN [1758](#), [1765](#), [1771](#), [1777](#), [1785](#), [1792](#),
 [1798](#), [1804](#), [3862](#)
 \int_gzero:N . [344](#), [345](#), [346](#), [1535](#), [1562](#), [2142](#), [2861](#),
 [3579](#), [3714](#), [4400](#), [4650](#)
 \int_if_exist:NnTF [1733](#), [1769](#), [1775](#), [1796](#), [1802](#), [1980](#)
 \int_incr:N [2616](#), [3447](#), [3635](#), [3791](#), [4179](#), [4242](#), [4433](#),
 [4473](#)
 \int_mod:nn [4391](#), [4640](#)
 \int_new:N . [28](#), [29](#), [30](#), [31](#), [32](#), [33](#), [61](#), [62](#), [86](#), [102](#), [121](#),
 [141](#), [142](#), [147](#), [148](#), [149](#), [151](#), [162](#), [168](#), [169](#), [170](#), [171](#),
 [172](#), [1735](#), [1983](#)

\int_set:Nn [741](#), [745](#), [747](#), [1872](#), [1879](#), [1891](#), [1900](#), [2753](#),
 [3836](#), [3837](#), [3892](#), [3923](#), [3946](#), [3952](#), [3968](#), [3995](#), [4001](#),
 [4017](#), [4371](#), [4620](#), [4818](#), [4956](#)
 \int_set_eq:NN [1860](#), [1865](#), [3975](#), [4024](#)
 \int_sign:n [2077](#)
 \int_step_function:nnN [2401](#), [2414](#), [2428](#)
 \int_step_function:nnnN [4960](#)
 \int_step_inline:nn [4870](#)
 \int_step_inline:nnn [3838](#)
 \int_to_roman:n [207](#), [2381](#), [2418](#)
 \int_use:N [366](#), [371](#), [372](#), [1191](#), [1210](#), [1522](#), [1874](#), [1881](#),
 [1893](#), [1902](#), [3346](#), [3365](#), [3390](#), [3467](#), [3532](#), [3541](#), [3556](#),
 [3562](#), [3950](#), [3951](#), [3963](#), [3999](#), [4000](#), [4012](#), [5291](#), [5295](#),
 [5301](#), [5305](#)
 \int_zero:N [4381](#), [4630](#)
 \item . [87](#), [90](#), [110](#), [112](#), [115](#), [117](#), [379](#), [2302](#), [2308](#), [2333](#), [2339](#),
 [2461](#), [2965](#), [2968](#), [3142](#), [3271](#), [3823](#), [4163](#), [4165](#), [4415](#),
 [4417](#), [4538](#)
 \item* [5](#), [14](#), [68](#), [3269](#)
 item-pos* [3167](#)
 item-sym* [3167](#)
 \itemindent [92](#)
 \itemindent [91](#)
 itemindent [873](#)
 \itemsep [101](#), [102](#)
 \itemsep [3807](#), [3813](#)
 \itemwidth . [455](#), [2152](#), [3428](#), [3437](#), [3614](#), [3623](#), [3986](#), [3990](#),
 [4035](#), [4039](#)

K

keyans [14](#), [3593](#)
 keyans* [14](#), [4404](#)
 keyanspic [15](#), [3767](#)
 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](#), [109](#)
 above [27](#), [57](#), [58](#), [96](#), [109](#), [115](#)
 after [45](#), [46](#), [97](#), [109](#), [115](#)
 align [27](#), [37](#), [88](#), [91](#), [112](#), [125](#)
 base-fix [43](#), [59](#), [71](#), [95](#), [109](#), [120](#), [121](#)
 before* [45](#), [46](#), [96](#), [109](#), [115](#)
 before [45](#), [46](#)
 below* [27](#), [57–59](#), [97](#), [109](#)
 below [27](#), [57–59](#), [97](#), [109](#), [115](#)
 check-ans . [29](#), [30](#), [32](#), [64–68](#), [71](#), [82](#), [85](#), [97](#), [98](#), [109](#), [113](#),
 [127](#)
 columns-sep [47](#), [96](#)
 columns [27](#), [47](#), [57](#), [96](#)
 first [45](#), [46](#), [112](#)
 font [37](#), [88](#), [91](#), [112](#)
 item-pos* [87](#), [88](#)
 item-sym* [28](#), [87](#), [88](#)
 itemindent [27](#), [44](#), [87](#), [91](#), [112](#)
 itemsep [42](#), [93](#)

labelsep	37, 92, 112
labelwidth	36–41, 92
label	26, 36, 38, 41, 104
lisparindent	93
list-indent	27, 44, 102
list-offset	44, 94, 98
listparindent	44, 112
mark-ans	69, 71, 76
mark-pos	69, 125
mark-ref	69, 71, 73, 75
mini-env	27, 34, 47, 56, 57, 71, 96, 106, 107, 109, 115
mini-right*	27, 30, 47, 71, 107, 109
mini-right	27, 30, 47, 55, 71, 107, 109
mini-sep	27, 47, 71, 96
no-store	29, 64–66, 71, 77, 87
noitemsep	42
nosep	42
parindent	93
parsep	42, 93, 112
partopsep	42
ref	26, 30, 38–40, 127
resume*	26, 59, 60, 63–65, 71, 97, 109, 121
resume	26, 33, 59–65, 71, 97, 109, 121
rightmargin	44, 104
save-ans	28, 33, 60–64, 66, 67, 70–72, 77–79, 82–84, 90, 98, 100, 114, 115, 117, 119, 121, 127
save-key	28, 60, 70, 95, 109
save-pos	71
save-ref	29, 35, 69, 71, 73–75, 83, 85, 90, 117
save-sep	69, 71, 116
series	26, 59–63, 71, 95, 97, 109, 121
show-ans	69, 71, 73, 74, 76, 90, 117
show-length	31, 45, 126
show-pos	28, 69, 73, 74, 76, 85, 90, 117
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, 111, 112, 116
wrap-label	27, 37, 87, 88, 91, 111, 112, 116
wrap-opt	69, 71
keys commands:	
\keys_define:nn	507, 529, 561, 577, 624, 695, 769, 795, 839, 875, 898, 971, 980, 1059, 1076, 1573, 1684, 1927, 1988, 2147, 2189, 2225, 2230, 2544, 2695, 2731, 3169, 3185, 3205, 3225, 4681, 4780, 4896, 4904
\keys_if_exist_p:nn	4892, 4893
\l_keys_key_str	77, 80, 2562, 2717, 3193, 3211, 3233, 4939, 5047
\keys_precompile:nnN	120, 191, 191, 4683, 4687, 4691, 4695, 4699, 4703, 4922
\keys_set:nn	521, 855, 866, 1082, 1578, 1583, 1821, 1826, 1913, 1921, 2582, 3461, 3466, 3651, 4197, 4455, 4735, 4742, 4784, 4789, 4790, 4791, 4792, 4795, 4800, 4801, 4802, 4803, 4804, 4805, 4806, 4838, 4948
\keys_set_known:nn	2849
keyval commands:	
\keyval_parse:NNn	1698, 2255, 4756
L	
label	575, 622, 695
Labels provide by enumext:	
\Alph*	36
\Roman*	36

\alph*	36
\arabic*	30, 36
\roman*	36
\labelsep	102
\labelsep	3808, 3811
labelsep	505
\labelwidth	36, 102
\labelwidth	3808, 3809
labelwidth	505
\lastkern	242
\lastnodetype	231
\lastskip	236
\leftmargin	92
\leftmargin	91, 3808
legacy commands:	
\legacy_if:nTF	4309, 4312, 4564, 4567
\legacy_if_gset_false:n	393
\legacy_if_set_false:n	4311, 4566
\legacy_if_set_true:n	4271, 4296, 4303, 4316, 4502, 4530, 4571
\linewidth	96
\linewidth	3430, 3518, 3616, 3661, 3835, 3895, 3926, 4048, 4103
\list	377
list-indent	873
list-offset	873
\listparindent	3810
listparindent	873
M	
\makebox	104
\makebox	2360, 2362, 3136, 4345, 4353, 4357, 4594, 4598
\makelabel	87, 88, 91, 103
\makelabel	87, 90, 3153, 3287
\makesavenoteenv	426
mark-ans	2145
mark-pos	2145, 2187
mark-ref	2145
mini-env	1057
mini-sep	1057
\minipage	383
\miniright	10, 55, 1491, 1539, 1566, 3577, 3712
mode commands:	
\mode_if_math:TF	2625, 2679
\mode_if_vertical:TF	1127, 1156, 1177, 1201, 1352, 1373
\mode_leave_vertical:	853, 864, 926, 940, 952, 964, 2358, 3134, 4343
msg commands:	
\msg_error:nn	1541, 1568, 2586, 2619, 2623, 2677, 2785, 3642, 3646, 3734, 3794, 3825, 4182, 4436, 4448, 4807, 4866
\msg_error:nnn	600, 647, 664, 717, 1495, 1502, 1507, 1537, 1564, 1833, 1837, 1952, 2568, 2627, 2645, 2657, 2665, 2669, 2673, 2681, 2723, 3199, 3217, 3239, 4186, 4441, 4669, 4678, 4749, 4854, 4885, 4894, 4931, 4952
\msg_error:nnnn	2571, 2599, 2603, 2607, 2611, 2726, 3202, 3220, 3242, 3633, 3730, 3738, 4431, 4730, 4934
\msg_error:nnnnn	548, 568, 2168
\msg_fatal:nn	3449
\msg_fatal:nnn	459
\msg_info:nnn	13, 16, 21, 24, 408, 422
\msg_line_context:	5012, 5017, 5022, 5051, 5056, 5061, 5076, 5091, 5095, 5099, 5103, 5107, 5111, 5118, 5125, 5131, 5145, 5149, 5154, 5158, 5162, 5166, 5171,

5175, 5179, 5183, 5188, 5223, 5227, 5232, 5237, 5241, 5246, 5322, 5326, 5331, 5336, 5341, 5345, 5349, 5353, 5357, 5361, 5365, 5369, 5373	
\msg_log:nnn	1972, 1977, 1982
\msg_log:nnnnn	370, 2115, 2120, 2125
\msg_log:nnnnnn	362
\msg_new:nnn 4979, 4983, 4987, 4991, 4996, 5009, 5014, 5019, 5024, 5033, 5041, 5045, 5049, 5054, 5059, 5074, 5089, 5093, 5097, 5101, 5105, 5109, 5113, 5122, 5128, 5134, 5138, 5142, 5147, 5152, 5156, 5160, 5164, 5169, 5173, 5177, 5181, 5186, 5221, 5225, 5230, 5235, 5239, 5244, 5320, 5324, 5329, 5334, 5339, 5343, 5347, 5351, 5355, 5359, 5363, 5367, 5371	
\msg_new:nnnn	5000, 5191, 5200, 5209, 5215, 5248, 5258, 5268, 5278, 5288, 5298, 5308, 5314
\msg_term:nnnn	1936, 1941, 3355, 3365, 3396, 3401
\msg_term:nnnnn	2096
\msg_warning:nn	3576, 3711
\msg_warning:nnnn 2133, 2139, 3303, 3308, 3949, 3962, 3998, 4011	
\msg_warning:nnnnn	2091, 2101
\multicolsep	96
\multicolsep	1194, 1366, 3552, 3688
N	
\NeedsTeXFormat	3
\newcounter	462
\NewDocumentCommand 1491, 2574, 3726, 4653, 4708, 4814, 4863, 4941	
\NewDocumentEnvironment	3406, 3593, 3767, 4151, 4404
\newenvsc	2688
\newlabel	35
\newlabel	444
no-store	1986
\noindent	4057, 4112, 4164, 4380, 4416, 4629
\nointerlineskip 1203, 1206, 1375, 1378, 1529, 1556, 4057, 4112	
noitemsep	793
\nopagebreak 1139, 1168, 1203, 1206, 1375, 1378, 1482, 1488	
\normalfont	2533, 3056, 3069, 4550
nosep	793
P	
Packages:	
caption	107
enumext	25, 35, 38, 64, 92, 100, 125
enumitem	36
expl3	103
footnotehyper	35
hyperref	29, 30, 34, 35, 75, 85, 112, 125
lua-visual-debug	50
multicol	25, 125
scontents	25, 78, 79
shortlst	103, 108, 112
\par	1139, 1168, 1206, 1378, 1482, 1488, 1524, 1529, 1551, 1556, 2510, 3567, 3702, 3717, 3847, 4075, 4089, 4130, 4144, 4380, 4629
para commands:	
\para_end:	4397, 4647
\parbox	2152
\parindent	4363, 4604
\parsep	48, 101, 102
\parsep	3387, 3807, 3814, 3819
parsep	793
\parskip	4364, 4605
\partopsep	102
\partopsep	3388, 3812
partopsep	793
peek commands:	
\peek_meaning:NNTF 4248, 4262, 4279, 4290, 4479, 4493, 4510	
\peek_meaning_remove:NNTF	4255, 4486
\peek_remove_spaces:n	3275
\phantomsection	35
\phantomsection	433
prg commands:	
\prg_do_nothing:	437
\prg_new_protected_conditional:Npnn	209
\prg_replicate:nn	226
\prg_return_false:	213
\prg_return_true:	212
\printkeyans	16, 119, 4708
prop commands:	
\prop_const_from_keyval:Nn	4855
\prop_count:N 364, 2284, 2435, 2536, 2946, 3059, 3072, 4553, 4957	
\prop_get:NnNTF	4881
\prop_gput_if_not_in:Nnn	2282
\prop_if_exist:NNTF	1970, 4673, 4950
\prop_item:Nn	4675, 4974
\prop_new:N	1973
\ProvidesExplPackage	4
R	
\raggedcolumns	3555, 3691
\ref	73, 83
ref	575, 622, 695
\refstepcounter	4318, 4573
regex commands:	
\regex_match:nnTF	211, 744, 746, 758, 760, 2781
\regex_replace_once:nnN	219
\renewcommand	610, 655, 672, 725
\RenewDocumentCommand 1539, 1566, 3142, 3153, 3271, 3287, 3823, 3857	
\RequirePackage	17, 25
resume	1682
resume*	1682
rightmargin	873
\Roman	36, 41
\Roman	481
\roman	36, 41
\roman	482, 593, 4698
S	
\s	2782
save-ans	1925
save-key	2223
save-ref	2145
save-sep	2145
scan commands:	
\scan_stop:	102, 3821, 4163, 4415, 4664, 4667
scontents internal commands:	
\l_scontents_fname_out_tl	2741
_scontents_parse_environment_keys:n	2747
_scontents_rescan_tokens:n	2754
\l_scontents_storing_bool	2739
\l_scontents_writing_bool	2740
seq commands:	
\seq_clear:N	4816, 4959
\seq_const_from_clist:Nn	4809
\seq_count:N	365, 3780, 4820

©2024 by Pablo González L

<code>\tl_put_left:Nn</code>	2306, 2337, 2459, 2801, 2814, 2820, 2830, 3038, 3078, 4078, 4133, 4535, 4538
<code>\tl_put_right:Nn</code>	492, 608, 653, 670, 723, 2310, 2341, 2388, 2398, 2411, 2426, 2432, 2437, 2461, 2466, 2473, 2476, 2486, 2491, 2494, 2500, 2891, 2894, 2901, 2903, 2930, 2935, 2940, 2943, 2952, 2965, 2968, 2974, 2979, 2989, 4521, 4522
<code>\tl_remove_all:Nn</code>	4845
<code>\tl_remove_once:Nn</code>	2376, 2915
<code>\tl_replace_all:Nnn</code>	496, 4880
<code>\tl_reverse:N</code>	2375, 2377, 2914, 2916
<code>\tl_set:Nn</code>	58, 253, 263, 312, 313, 320, 321, 328, 329, 461, 535, 539, 544, 545, 597, 642, 714, 924, 938, 950, 962, 1845, 1946, 2210, 2220, 2241, 2249, 2530, 2741, 3008, 3053, 3066, 4524, 4547, 4843, 4879, 4949
<code>\tl_set_eq:NN</code>	502, 603, 606, 650, 652, 667, 669, 720, 722, 2374, 2913, 2926, 3259, 3263, 3759, 3761
<code>\tl_to_str:n</code>	1816, 1822, 1827, 4656
<code>\tl_trim_spaces:n</code>	492, 4832, 4843, 4849, 4865
<code>\tl_use:N</code>	498, 501, 619, 684, 691, 734, 995, 999, 1003, 1007, 1011, 1015, 1019, 1023, 1027, 1031, 1035, 1039, 1043, 1047, 1051, 1055, 2364, 2381, 2389, 2400, 2413, 2418, 2429, 3094, 3100, 3128, 3155, 3156, 3163, 3250, 3254, 3262, 3289, 3290, 3296, 3413, 3599, 3764, 4085, 4140, 4350, 4361, 4365, 4591, 4602, 4608, 4613, 4711, 4712, 4713, 4714, 4715, 4733, 4828, 4947
token commands:	
<code>\token_to_str:N</code>	444
<code>topsep</code>	793
<code>\topskip</code>	1193, 1365
<code>\typeout</code>	235, 236, 241, 242, 412, 415, 425, 426
U	
<code>\u</code>	220, 2782
<code>\unkern</code>	243
<code>unknown</code>	3183, 3205, 3223
<code>\unskip</code>	237
use commands:	
<code>\use:N</code>	227, 3160, 3415
<code>\use:n</code>	1696, 2253, 4662, 4754
<code>\use_none:nn</code>	436, 4886
<code>\usecounter</code>	3345, 3389
V	
<code>\value</code>	1759, 1765, 1772, 1778, 1786, 1792, 1799, 1805
vbox commands:	
<code>\vbox_set_top:Nn</code>	4083, 4138
<code>\vspace</code>	854, 865, 1595, 1598, 1609, 1612, 1622, 1624, 1633, 1635, 1644, 1646, 1655, 1657, 1666, 1668, 1677, 1679, 3775, 3786
W	
<code>widest</code>	767
<code>wrap-ans</code>	2145
<code>wrap-label</code>	505
<code>wrap-label*</code>	505
<code>wrap-opt</code>	2145
Z	
<code>\z</code>	2782