

enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-09-23^{*}

©2024 by Pablo González[†]

CTAN: <https://www.ctan.org/pkg/enumext>

 <https://github.com/pablgonz/enumext>

Abstract

This package provides “*enumerated list*” environments for creating “*simple exercise sheets*” along with “*multiple choice questions*”, storing the `\answers` to these in memory using `multicol` and `scontents` packages and the `l3seq` and `l3prop` modules.

Contents

1	Introduction	1	6	The storage system	11
1.1	Description and usage	2	6.1	Keys for storage system	11
1.2	The concept of left margin	3	6.1.1	Keys for label and ref	11
1.3	User interface	3	6.1.2	Keys for wrap and display	12
1.3.1	Internal counters	3	6.1.3	Keys for debug and checking	12
1.3.2	Public dimension	3	6.2	The command <code>\anskey</code>	12
1.3.3	Support for <code>multicol</code>	3	6.2.1	Keys for <code>\anskey</code>	12
1.3.4	Support for <code>minipage</code>	4	6.3	The environment <code>anskey*</code>	13
1.3.5	The <code>\label</code> and <code>\ref</code> system	4	6.3.1	Keys for <code>anskey*</code>	13
1.3.6	Support for <code>\footnote</code>	4	6.4	The environment <code>keyans</code>	14
2	The environments provided	4	6.4.1	The <code>\item*</code> in <code>keyans</code>	14
2.1	The environment <code>enumext</code>	4	6.5	The environment <code>keyanspic</code>	15
2.2	The environment <code>enumext*</code>	5	6.5.1	The command <code>\anspic</code>	15
2.3	The command <code>\item*</code>	5	6.6	Printing stored content	16
2.3.1	Keys for <code>\item*</code>	5	6.6.1	The command <code>\getkeyans</code>	16
2.4	The command <code>\item</code> in <code>enumext*</code>	5	6.6.2	The command <code>\foreachkeyans</code>	16
3	The command <code>\setenumext</code>	6	6.6.3	The command <code>\printkeyans</code>	16
4	The command <code>\setenumextmeta</code>	6	7	Full examples	17
5	The <code>keyval</code> system	6	8	The way of non-enumerated lists	20
5.1	Keys for label and ref	6	9	References	22
5.2	Keys for spaces	7	10	Change history	22
5.2.1	Vertical spaces	7	11	Index of Documentation	23
5.2.2	Horizontal spaces	8	12	Implementation	25
5.3	Keys for add code	9	13	Index of Implementation	137
5.4	Keys for start, series and resume	9			
5.5	Keys for <code>multicols</code>	10			
5.6	Keys for <code>minipage</code>	10			
5.6.1	The command <code>\miniright</code>	10			
5.6.2	The key <code>mini-right</code>	10			

Motivation and acknowledgments

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

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

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

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

^{*}This file describes a documentation for v1.0, last revised 2024-09-23.

[†]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

C) value

B) correct

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

D) I and III only

B) II only

E) I, II, and III

C) I and II only

★ 3. Third type of questions

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

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

A) value

D) value

B) value

E) value

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

A

A)

B

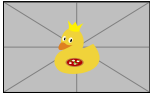
B)

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 / 152

1. B), $x = 5$

2. D)

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

* 5. D), “other note”

*

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

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

1.1 Description and usage

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

- This package is NOT intend to replace the `enumerate` environment nor replace the powerful `enumitem`[6], the approach is intended to work without hindering either of them.

This package can be used with `xelatex`, `lualatex`, `pdflatex` and the classical `latex»dvips»ps2pdf` and is present in \TeX Live and \MiKTeX , use the package manager to install. For manual installation, download `enumext.zip` and unzip it, run `lualatex enumext.dtx` and move all files to appropriate locations, then run `mktexlsr`. To produce the documentation run `lualatex enumext.dtx` two times.

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

The package is loaded in the usual way:

```
\usepackage{enumext}
```

1.2 The concept of left margin

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

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

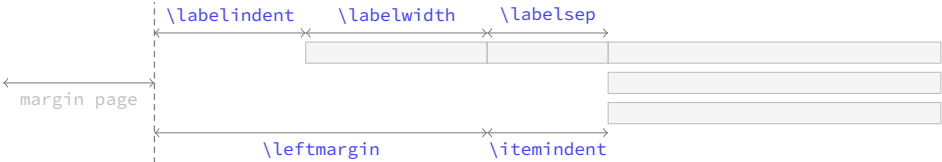


Figure 1: Representation of horizontal lengths in `enumitem`.

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

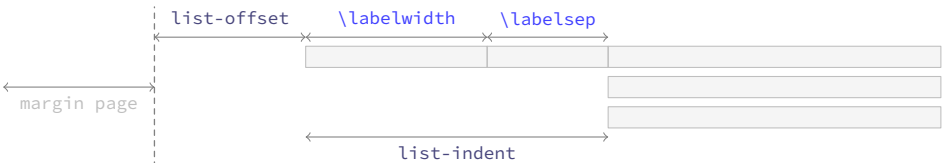


Figure 2: Representation of horizontal lengths concept in `enumext`.

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

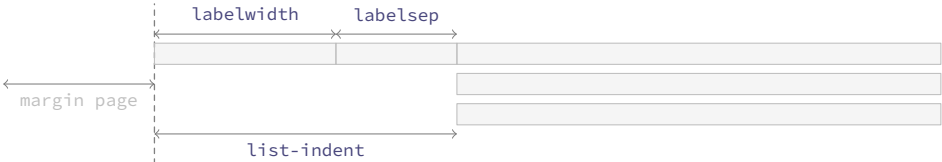


Figure 3: Default horizontal lengths `list-offset=0pt`, `list-indent=\labelwidth+\labelsep` in `enumext`.

1.3 User interface

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

1.3.1 Internal counters

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

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

1.3.2 Public dimension

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

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

1.3.3 Support for multicol

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



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

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

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

1.3.4 Support for minipage

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



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

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

1.3.5 The \label and \ref system

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

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

1.3.6 Support for \footnote

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

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

2 The environments provided

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

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

2.1 The environment enumext

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

Example with `columns=2`

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

2.2 The environment enumext*

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

Some considerations to take into account for this environment:

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

Example with `columns=2`

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

2.3 The command \item*

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

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

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

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

2.3.1 Keys for `\item*`

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

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

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

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

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

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

3 The command `\setenumext`

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

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

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

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

4 The command `\setenumextmeta`

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

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

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

5 The keyval system

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

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

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

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

5.1 Keys for label and ref

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

default: *empty*

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

5.2 Keys for spaces

`show-length = {⟨true | false⟩}`

default: *false*

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

5.2.1 Vertical spaces

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

default: *by levels*

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

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

default: *by levels*

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

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

default: *by levels*

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

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

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

default: *by levels*

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

`noitemsep` *⟨value forbidden⟩*

default: *not used*

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

`nosep` *⟨value forbidden⟩*

default: *not used*

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

`base-fix` *⟨value forbidden⟩*

default: *not used*

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

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

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

default: *not used*

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

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

default: *not used*

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

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

default: *not used*

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

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

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

5.2.2 Horizontal spaces

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

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

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

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

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

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

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

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

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

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

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

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

5.3 Keys for add code

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

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

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

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

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

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

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

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

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

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

5.4 Keys for start, series and resume

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

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

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

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

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

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

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

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

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

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

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

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

5.5 Keys for multicols

`columns = {⟨integer⟩}` default: `1`

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

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

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

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

5.6 Keys for minipage

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

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

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

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

5.6.1 The command `\miniright`

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

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

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

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

5.6.2 The key `mini-right`

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

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

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

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

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

6 The storage system

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

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

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

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

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

6.1 Keys for storage system

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

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

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

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

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

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

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

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

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

6.1.1 Keys for label and ref

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

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

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

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

6.1.2 Keys for wrap and display

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

6.1.3 Keys for debug and checking

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

6.2 The command `\anskey`

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

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

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

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

6.2.1 Keys for `\anskey`

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

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

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

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

Example

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

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

6.3 The environment `anskey*`

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

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

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

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

6.3.1 Keys for `anskey*`

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

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

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

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

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

Example

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

```

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

```

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

6.4 The environments `keyans` and `keyans*`

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

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

<pre> \begin{enumext}[save-ans=test] \item \langle item content \rangle \begin{keyans}[\langle key = val \rangle] \item \langle item content \rangle \item [\langle custom \rangle] \langle item content \rangle \item* \langle item content \rangle \item*[\langle content \rangle] \langle item content \rangle \end{keyans} \end{enumext} </pre>	<pre> \begin{enumext}[save-ans=test] \item \langle item content \rangle \begin{keyans*}[\langle key = val \rangle] \item \langle item content \rangle \item [\langle custom \rangle] \langle item content \rangle \item* \langle item content \rangle \item*[\langle content \rangle] \langle item content \rangle \end{keyans*} \end{enumext} </pre>
---	---

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

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

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

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

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

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

Example


```

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

```



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

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

6.5 The environment keyanspic

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

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



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

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

6.5.1 The command \anspic

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

The `\anspic` command take three arguments, the *starred argument* ‘*’ store the current `\label` next to the `\content` (if it is present) in *sequence* and *prop list* `{\store name}` set by `save-ans` key. The *starred argument* ‘*’ cannot be separated by spaces ‘ ’ from the command, i.e. `\anspic*` and the optional argument does “not support” verbatim content. By design it is assumed that the *starred argument* ‘*’ will only appear “once” within the environment.

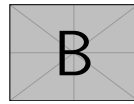
Example

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

1. Question with images.



A)



B)



C)



D)



* E)[note]

6.6 Printing stored content

6.6.1 The command `\getkeyans`

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

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

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

6.6.2 The command `\foreachkeyans`

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

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

Options for command

- `sep = {code}` default: empty
Establishes the separation between *each* content stored in *prop list* `{store name}`. For example, you can use `sep={\ [10pt]}` for vertical separation of stored contents.
- `step = {integer}` default: 1
Sets the increment (`{step}`) applied to the value set by key `start` for each element stored in *prop list* `{store name}`. The value must be a *positive integer*.
- `start = {integer}` default: 1
Sets the *position* of the *prop list* `{store name}` from which execution will start. The value must be a *positive integer*.
- `stop = {integer}` default: 0
Sets the *position* of the *prop list* `{store name}` from which execution it will finish executing. The value must be a *positive integer*.
- `before = {code}` default: empty
Sets the `{code}` that will be executed *before* each content stored in *prop list* `{store name}`. The `{code}` must be passed between braces.
- `after = {code}` default: empty
Sets the `{code}` that will be executed *after* each content stored in *prop list* `{store name}`. The `{code}` must be passed between braces.
- `wrapper = {code {#1} more code}` default: empty
Wraps the content stored in *prop list* `{store name}` referenced by `{#1}`. The `{code}` must be passed between braces. For example `\foreachkeyans wrapper={\makebox[1em][l]{#1}}{store name}`.

6.6.3 The command `\printkeyans`

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

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

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

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

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

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

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

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

Example

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

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

  \item Related to Linux

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

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

```
\printkeyans{sample}
```

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

[1] $3(x + y + z)$

2. True False

(a) ~~LaTeXe~~ is cool?

[2] Very True!

3. Related to Linux

(a) You use linux?

[3] Yes

(b) Rate the following package and class

i. `xsim`

[4] very good

ii. `exsheets`

[5] obsolete

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

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

2. (a) Very True!

3. (a) Yes

(b) i. very good

ii. obsolete
- *

*

*

*

*

7 Full examples

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

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


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

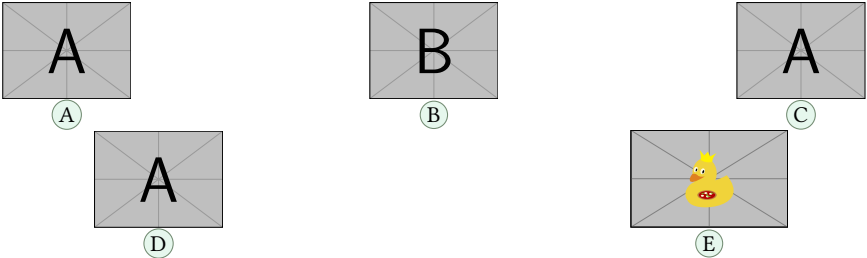
(2) $\angle EDF = 45^\circ$

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) ~~TeX~~ze is cool?

Related to Linux

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

ii. `xsim`

iii. `exsheets`

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

1. $(x - 1)^2$

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

Which choice best describes what happens in the

- passage?
- A) One character argues with another character who intrudes on her home.
 - B) One character receives a surprising request from another character.
 - C) One character reminisces about choices she has made over the years.
 - D) One character criticizes another character for pursuing an unexpected course of action.

3

Which choice best describes what happens in the passage?

- A) One character argues with another character who intrudes on her home.
- B) One character receives a surprising request from another character.

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

4

Which choice best describes what happens in the passage?

- A) One character argues with another character who intrudes on her home.
- B) One character receives a surprising request from another character.
- C) One character reminisces about choices she has made over the years.
- D) One character criticizes another character for pursuing an unexpected course of action.

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

8 The way of non-enumerated lists

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

Fake itemize environment

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

- First level item
 - Second level item
 - * Third level item
 - Fourth level item
 - First level item
- * First level item
 - ◊ Second level item
 - Third level item
 - ★ Fourth level item
 - * First level item


Fake description environment

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

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

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

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

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

Description indented by label

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

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

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

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

Final notes

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

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

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

Why list environments?

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

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

Why not random questions and other utilities

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

9 References

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

10 Change history

v1.0 2024-09-23 – First public release.

11 Index of Documentation

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

C

Document class:

article 2

book 2

exam 2

letter 2

report 2

\columnbreak 4, 12

\columnsep 10

Commands provide by enumext:

\anskey 11–13

\anspic 11, 12, 15

\foreachkeyans 16

\getkeyans 12, 16

\item* 5–7, 11, 12, 14, 15

\item 5–10, 12, 14

\miniright 10

\printkeyans 6, 11, 16

\setenumextmeta 6

\setenumext 5–7, 11, 12, 14, 17

Counters defined by enumext:

enumXiii 4

enumXii 4

enumXiv 4

enumXi 4

enumXviii 4

enumXvii 4

enumXvi 4

enumXv 4

E

Environments provide by enumext:

anskey* 11–13

enumext* 4–14, 16, 17

enumext 4–14, 16, 17, 20

keyans* 4–14

keyanspic 4, 7, 8, 11–13, 15, 20

keyans 4–9, 11–15, 20

Environments:

Verbatim 13

enumerate 1, 3, 5, 21

figure 5

list 3, 9, 21

minipage 3–5, 10, 21

multicols 3, 4, 10

table 5

task 5

F

\footnote 5

I

\itemsep 8

K

Keys for \anskey provide by enumext:

break-col 12

item-join 12

item-pos* 13

item-star 12, 13

item-sym* 13

Keys for \foreachkeyans provide by enumext:

after 16

before 16

sep 16

start 16

step 16

stop 16

wrapper 16

Keys for anskey* provide by enumext:

break-col 12

force-eol 13

item-join 12

item-pos* 13

item-star 12, 13

item-sym* 13

overwrite 13

write-env 13

Keys for environments provide by enumext:

above* 8

above 8

after 9, 10

align 7, 21

base-fix 8

before* 9

before 9

below* 9

below 8

check-ans 12

columns-sep 4, 10

columns 4, 8, 10

first 9

font 7

item-pos* 5, 6

item-sym* 5, 6

itemindent 9

itemsep 8, 15

labelsep 3–7, 9, 10, 12, 20, 21

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

labelwith 5

label 7, 9, 14, 20, 21

list-indent 3, 9

list-offset 3, 9, 21

listparindent 9

mark-ans 12

mark-pos 12

mark-ref 11

mini-env 4, 5, 8, 10

mini-right* 7, 10

mini-right 7, 10

mini-sep 4, 10

no-store 11–13

noitemsep 8

nosep 8, 20

overwrite 13

parsep 8, 15

partopsep 8

ref 4, 7

resume* 7, 10, 11

resume 7, 10, 11

rightmargin 9

save-ans 4, 6, 10–16

©2024 by Pablo González L

23 / 152

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

12 Implementation

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

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

12.1 General conventions

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

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

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

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

12.2 Initial set up

Start the DocStrip guards.

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

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

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

12.3 Declaration of the package

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

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

Now declare the `enumext` package.

```
4 \ProvidesExplPackage
5   {enumext}
6   {2024-09-23}
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
\l__enumext_align_label_pos_X_str
```

Internal variables used by `columns` key (§12.21) and `align` key (§12.12).

```
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   \str_new:c { \l__enumext_align_label_pos_#1_str }
85 }
86 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }
```

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

```
\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).

```
87 \int_new:N \g__enumext_minipage_stat_int
88 \skip_new:N \l__enumext_minipage_left_skip
89 \skip_new:N \l__enumext_minipage_right_skip
90 \skip_new:N \l__enumext_minipage_after_skip
91 \skip_new:N \g__enumext_minipage_right_skip
92 \skip_new:N \g__enumext_minipage_after_skip
93 \cs_set_protected:Npn \__enumext_tmp:n #1
94 {
95   \dim_new:c { \l__enumext_minipage_left_#1_dim }
96   \bool_new:c { \l__enumext_minipage_active_#1_bool }
97 }
98 \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).

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

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

```

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

```

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

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

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

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

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

```

111 \bool_new:N \l__enumext_store_active_bool
112 \tl_new:N \l__enumext_store_name_tl
113 \tl_new:N \g__enumext_store_name_tl
114 \tl_new:N \l__enumext_store_anskey_arg_tl
115 \tl_new:N \l__enumext_store_anskey_env_tl
116 \tl_new:N \l__enumext_store_anskey_opt_tl
117 \tl_new:N \l__enumext_store_current_label_tl
118 \tl_new:N \l__enumext_store_current_opt_arg_tl
119 \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).

```

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

```

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

```

\l__enumext_meta_path_tl
\l__enumext_foreach_print_seq
\l__enumext_foreach_name_prop_tl
\g__enumext_foreach_default_keys_tl

```

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

```

125 \tl_new:N \l__enumext_meta_path_tl
126 \seq_new:N \l__enumext_foreach_print_seq
127 \tl_new:N \l__enumext_foreach_name_prop_tl
128 \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*”.

```

129 \tl_new:N \l__enumext_print_keyans_starred_tl
130 \str_new:N \l__enumext_mark_position_str
131 \tl_new:N \g__enumext_item_symbol_aux_tl
132 \cs_set_protected:Npn \__enumext_tmp:n #1
133 {
134   \tl_new:c { \l__enumext_print_keyans_#1_tl }
135   \tl_new:c { \l__enumext_store_save_key_#1_tl }
136   \bool_new:c { \l__enumext_store_save_key_#1_bool }
137   \bool_new:c { \l__enumext_store_upper_level_#1_bool }
138 }
139 \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_star_bool
\l__enumext_keyans_pic_label_pos_str
\g__enumext_keyans_pic_parsep_skip
\l__enumext_anspic_label_box
\l__enumext_anspic_body_box
\l__enumext_anspic_label_htdp_dim
\l__enumext_anspic_body_htdp_dim

```

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

```

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

```

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

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

```

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

```

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

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

```

159 \bool_new:N \l__enumext_hyperref_bool
160 \bool_new:N \l__enumext_footnotes_key_bool

```

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

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

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

```

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

```

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

Internal variables used for redefinition of `\footnote` (§12.42.4).

```

169 \int_new:N \g__enumext_footnote_int
170 \seq_new:N \g__enumext_footnote_arg_seq
171 \seq_new:N \g__enumext_footnote_int_seq

```

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

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

```

172 \cs_set_protected:Npn \__enumext_tmp:n #1
173 {
174   \bool_new:c { l__enumext_item_starred_#1_bool }
175   \int_new:c { l__enumext_item_column_pos_#1_int }
176   \int_new:c { g__enumext_item_count_all_#1_int }
177   \int_new:c { l__enumext_joined_item_#1_int }
178   \int_new:c { l__enumext_joined_item_aux_#1_int }
179   \int_new:c { l__enumext_tmpa_#1_int }
180   \dim_new:c { l__enumext_tmpa_#1_dim }
181   \box_new:c { l__enumext_item_text_#1_box }
182   \dim_new:c { l__enumext_joined_width_#1_dim }
183   \dim_new:c { l__enumext_item_width_#1_dim }
184   \tl_new:c { g__enumext_item_symbol_aux_#1_tl }
185   \str_new:c { l__enumext_align_label_#1_str }
186   \bool_new:c { g__enumext_minipage_active_#1_bool }
187   \box_new:c { l__enumext_miniright_code_#1_box }
188   \bool_new:c { g__enumext_minipage_center_#1_bool }
189   \dim_new:c { g__enumext_minipage_right_#1_dim }
190   \skip_new:c { g__enumext_minipage_right_#1_skip }
191 }
192 \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
193 \clist_const:Nn \c__enumext_all_envs_clist
194 {
195     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
196     {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
197 }
```

(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
198 \cs_generate_variant:Nn \keys_precompile:nnN { neN }
199 \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.

```
200 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
201 {
202     \hook_gput_code:nnn {begindocument} {enumext} { #1 }
203 }
```

(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
204 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
205 {
206     \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
207 }
208 \cs_new_protected:Npn \__enumext_before_env:nn #1 #2
209 {
210     \hook_gput_code:nnn {env/#1/before} {enumext} {#2}
211 }
```

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

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

```
212 \cs_new:Nn \__enumext_level:
213 {
214     \int_to_roman:n { \__enumext_level_int }
215 }
```

(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
216 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
217 {
218     \regex_match:nnTF { ^[\+|-]?[\d]+$ } {#1} % $
219     { \prg_return_true: }
220     { \prg_return_false: }
221 }
```

(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.

```
222 \cs_new_protected:Nn \__enumext_regex_counter_style:
223 {
224     \tl_map_inline:Nn \c__enumext_counter_style_tl
225     {
226         \regex_replace_once:nnN { \c{##1}\* }
227         { \c{##1}\cB{\u{\l__enumext_ref_the_count_tl}\cE} } \l__enumext_ref_key_arg_tl
228     }
229 }
```

(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.

```

230 \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
231 {
232     * ~ #2
233     \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
234     = ~ \use:c { #1_use:c } { \__enumext_#2_#3_#1 } \\
235 }

```

(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.

```

236 \cs_new_protected:Npn \__enumext_unskip_unkern:
237 {
238     \int_case:nnT { \lastnodetype }
239     {
240         { 11 }
241         {
242             \typeout{SKIIIIIIIIIIIIIIP}
243             \typeout{\the\lastskip}
244             \unskip
245         }
246         { 12 }
247         {
248             \typeout{KERRRRRRRRRRRRRRRN}
249             \typeout{\the\lastkern}
250             \unkern
251         }
252     }
253 }

```

(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:`

```

254 \cs_new_protected:Nn \__enumext_is_not_nested:
255 {
256     \str_case:en { \@currenvir }
257     {
258         {enumext}
259         {
260             \tl_set:Nn \l__enumext_envir_name_tl { enumext }
261             \bool_lazy_and:nnT
262             { \bool_not_p:n { \g__enumext_standar_bool } }
263             { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
264             {
265                 \bool_gset_true:N \g__enumext_standar_bool
266             }
267         }
268         {enumext*}
269         {
270             \tl_set:Nn \l__enumext_envir_name_tl { enumext* }
271             \bool_lazy_and:nnT
272             { \bool_not_p:n { \g__enumext_starred_bool } }
273             { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
274             {
275                 \bool_gset_true:N \g__enumext_starred_bool
276             }
277         }
278     }
279 }

```

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.

```

280 \cs_new_protected:Nn \__enumext_is_on_first_level:
281 {
282   \bool_lazy_all:nT
283   {
284     { \bool_if_p:N \g__enumext_standar_bool }
285     { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
286     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
287   }
288   {
289     \bool_set_true:N \l__enumext_standar_first_bool
290     \bool_set_true:N \l__enumext_anskey_env_bool
291     \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
292     \tl_gset:Ne \g__enumext_start_line_tl
293     {
294       on ~ line ~ \exp_not:V \inputlineno
295     }
296   }
297   \bool_lazy_all:nT
298   {
299     { \bool_if_p:N \g__enumext_starred_bool }
300     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
301     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
302   }
303   {
304     \bool_set_true:N \l__enumext_starred_first_bool
305     \bool_set_true:N \l__enumext_anskey_env_bool
306     \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
307     \tl_gset:Ne \g__enumext_start_line_tl
308     {
309       on ~ line ~ \exp_not:V \inputlineno
310     }
311   }
312 }

```

(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.

```

313 \cs_new_protected:Nn \__enumext_keyans_name_and_start:
314 {
315   \str_case:en { \@currentenvir }
316   {
317     {keyans}
318     {
319       \tl_set:Nn \l__enumext_envir_name_tl { keyans }
320       \tl_set:Ne \l__enumext_check_start_line_env_tl
321       {
322         in ~ 'keyans' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
323       }
324     }
325     {keyans*}
326     {
327       \tl_set:Nn \l__enumext_envir_name_tl { keyans* }
328       \tl_set:Ne \l__enumext_check_start_line_env_tl
329       {
330         in ~ 'keyans*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
331       }
332     }
333     {keyanspic}
334     {
335       \tl_set:Nn \l__enumext_envir_name_tl { keyanspic }
336       \tl_set:Ne \l__enumext_check_start_line_env_tl
337       {
338         in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
339       }
340     }
341   }
342 }

```


(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:
343 \cs_new_protected:Nn \__enumext_reset_global_vars:
344 {
345   \__enumext_reset_global_int:
346   \__enumext_reset_global_bool:
347   \__enumext_reset_global_tl:
348 }
349 \cs_new_protected:Nn \__enumext_reset_global_int:
350 {
351   \int_gzero:N \g__enumext_item_number_int
352   \int_gzero:N \g__enumext_item_anskey_int
353   \int_gzero:N \g__enumext_item_answer_diff_int
354 }
355 \cs_new_protected:Nn \__enumext_reset_global_bool:
356 {
357   \bool_gset_false:N \g__enumext_check_ans_key_bool
358   \bool_gset_false:N \g__enumext_standar_bool
359   \bool_gset_false:N \g__enumext_starred_bool
360 }
361 \cs_new_protected:Nn \__enumext_reset_global_tl:
362 {
363   \tl_gclear:N \g__enumext_store_name_tl
364   \tl_gclear:N \g__enumext_start_line_tl
365   \tl_gclear:N \g__enumext_envir_name_tl
366 }
```

(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.

```
367 \cs_new_protected:Nn \__enumext_log_global_vars:
368 {
369   \msg_log:nneeee { enumext } { prop-seq-int-hook }
370   { \g__enumext_store_name_tl }
371   { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
372   { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
373   { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
374 }
```

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.

```
375 \cs_new_protected:Nn \__enumext_log_answer_vars:
376 {
377   \msg_log:nneee { enumext } { item-answer-hook }
378   { \int_use:N \g__enumext_item_number_int }
379   { \int_use:N \g__enumext_item_anskey_int }
380   { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int } }
381 }
```

(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.

```
382 \__enumext_at_begin_document:n
383 {
```

```

384 \cs_new_eq:NN \__enumext_start_list:nn \list
385 \cs_new_eq:NN \__enumext_stop_list: \endlist
386 \NewCommandCopy \__enumext_item_std:w \item
387 }

```

(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.

```

388 \__enumext_at_begin_document:n
389 {
390 \cs_new_eq:NN \__enumext_minipage:w \minipage
391 \cs_new_eq:NN \__enumext_endminipage: \endminipage
392 }

```

(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)

```

393 \cs_new_protected:Nn \__enumext_internal_mini_page:
394 {
395 \int_compare:nNt { \l__enumext_level_int } = { 0 }
396 {
397 \DeclareDocumentEnvironment{__enumext_mini_page}{ m }
398 {
399 \__enumext_minipage:w [ t ] { ##1 }
400 \legacy_if_gset_false:n { @minipage }
401 \skip_vertical:N \c_zero_skip
402 }
403 {
404 \skip_vertical:N \c_zero_skip
405 \__enumext_endminipage:
406 }
407 }
408 }

```

(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.

```

409 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
410 \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”.

```

411 \cs_new_protected:Nn \__enumext_after_hyperref:
412 {
413 \IfPackageLoadedTF { hyperref }
414 {
415 \msg_info:nnn { enumext } { package-load } { hyperref }
416 \bool_set_true:N \l__enumext_hyperref_bool
417 \IfHyperBoolean{hyperfootnotes}
418 {

```

```

419         \typeout{hyperfootnotes=true}
420         \bool_set_true:N \l__enumext_footnotes_key_bool
421     }
422     { \typeout{hyperfootnotes=false} }
423 }
424 { }

```

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`.

```

425 \bool_if:NT \l__enumext_footnotes_key_bool
426 {
427     \IfPackageLoadedTF { footnotehyper }
428     {
429         \msg_info:nnn { enumext } { package-load } { footnotehyper }
430     }
431     {
432         \typeout{No ~ footnotehyper ~ load}
433         \typeout{Load ~ and ~ use ~ \string\makesavenoteenv{enumext*}}
434         \bool_set_false:N \l__enumext_footnotes_key_bool
435     }
436 }

```

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.

```

437 \bool_if:NTF \l__enumext_hyperref_bool
438 {
439     \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
440     \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
441 }
442 {
443     \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
444     \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
445 }
446 }

```

(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.

```

447 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
448 {
449     \protected@write \@auxout { }
450     {
451         \token_to_str:N \newlabel {#1}
452         {
453             {#2}
454             \bool_if:NT \l__enumext_hyperref_bool
455             { { \thepage } {#2} {#1} }
456             { }
457         }
458     }
459     \__enumext_hypertarget:nn {#1} { }
460     \__enumext_phantomsection:
461 }

```

(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.

```

462 \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.

```
463 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
464 {
465   \cs_if_exist:cTF { c@ #2 }
466   { \msg_fatal:nnn { enumext } { counters }{ #2 } }
467   {
468     \tl_set:Nn #1 { #2 }
469     \newcounter { #2 }
470   }
471 }
```

(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      472 \__enumext_define_counters:Nn \__enumext_counter_i_tl { enumXi }
enumXii     473 \__enumext_define_counters:Nn \__enumext_counter_ii_tl { enumXii }
enumXiii    474 \__enumext_define_counters:Nn \__enumext_counter_iii_tl { enumXiii }
enumXiv     475 \__enumext_define_counters:Nn \__enumext_counter_iv_tl { enumXiv }
enumXv      476 \__enumext_define_counters:Nn \__enumext_counter_v_tl { enumXv }
enumXvi     477 \__enumext_define_counters:Nn \__enumext_counter_vi_tl { enumXvi }
enumXvii    478 \__enumext_define_counters:Nn \__enumext_counter_vii_tl { enumXvii }
enumXviii   479 \__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.

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

```
490 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
491 {
492   \hbox_set:Nn \l__enumext_label_width_by_box {#2}
493   \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
494 }
495 \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.

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

```

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

```

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

12.12 Setting keys associated with label

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

```

labelsep
labelwidth
wrap-label
wrap-label*
512 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
513 {
514   \keys_define:nn { enumext / #1 }
515   {
516     font .tl_set:c = { l__enumext_label_font_style_#2_tl },
517     font .value_required:n = true,
518     labelsep .dim_set:c = { l__enumext_labelsep_#2_dim },
519     labelsep .initial:n = {0.3333em},
520     labelsep .value_required:n = true,
521     labelwidth .dim_set:c = { l__enumext_labelwidth_#2_dim },
522     labelwidth .value_required:n = true,
523     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
524     wrap-label .initial:n = {##1},
525     wrap-label .value_required:n = true,
526     wrap-label* .code:n = {
527       \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
528       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
529     },
530     wrap-label* .value_required:n = true,
531   }
532 }
533 \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.

```

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

```

```

554         \tl_set:cn { \__enumext_label_fill_right_#2_tl } { \hfill }
555         \str_set:cn { \__enumext_align_label_pos_#2_str } { c }
556     },
557     align / unknown .code:n =
558         \msg_error:nneee { enumext } { unknown-choice }
559         { align } { left, ~ right, ~ center } { \exp_not:n {##1} },
560     align .initial:n = left,
561     align .value_required:n = true,
562 }
563 }
564 \clist_map_inline:nn
565 {
566     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
567 }
568 { \__enumext_tmp:nn #1 }

569 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
570 {
571     \keys_define:nn { enumext / #1 }
572     {
573         align .choice:,
574         align / left .code:n = \str_set:cn { \__enumext_align_label_#2_str } { l },
575         align / right .code:n = \str_set:cn { \__enumext_align_label_#2_str } { r },
576         align / center .code:n = \str_set:cn { \__enumext_align_label_#2_str } { c },
577         align / unknown .code:n =
578             \msg_error:nneee { enumext } { unknown-choice }
579             { align } { left, ~ right, ~ center } { \exp_not:n {##1} },
580         align .initial:n = left,
581         align .value_required:n = true,
582     }
583 }
584 \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

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

```

\l__enumext_label_i_tl
\l__enumext_label_ii_tl
\l__enumext_label_iii_tl
\l__enumext_label_iv_tl

585 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
586 {
587     \keys_define:nn { enumext / #1 }
588     {
589         label .code:n = {
590             \__enumext_label_style:cvn { \__enumext_label_#2_tl }
591             { \__enumext_counter_#2_tl } {##1}
592             \dim_set_eq:cN { \__enumext_labelwidth_#2_dim }
593             \l__enumext_current_widest_dim
594         },
595         label .initial:n = #3,
596         label .value_required:n = true,
597         ref .code:n = \__enumext_standar_ref:n {##1},
598         ref .value_required:n = true,
599     }
600 }
601 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
602 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*. ) }
603 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
604 \__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }

```

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

`__enumext_standar_ref:n` The `__enumext_standar_ref:n` first we will pass the key argument to `\l__enumext_ref_key_arg_tl` and we will analyze its state, if it is not *empty* we will make a copy of the current counter in `\l__enumext_ref_the_count_tl` and we will execute the function `__enumext_regex_counter_style:` which will

return the modified `\l__enumext_ref_key_arg_tl` and we make the value of `\l__enumext_ref_the_count_tl` the same as that `\l__enumext_the_counter_X_tl` which contains `\theenumX` and finally we set `\l__enumext_renew_the_count_X_tl` with the renewed command.

```

605 \cs_new_protected:Npn \__enumext_standar_ref:n #1
606 {
607   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
608   \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
609   {
610     \msg_error:nnn { enumext } { key-ref-empty } { enumext }
611   }
612   {
613     \tl_set_eq:Nc
614     \l__enumext_ref_the_count_tl { \l__enumext_counter_ \__enumext_level: _tl }
615     \__enumext_regex_counter_style:
616     \tl_set_eq:Nc
617     \l__enumext_ref_the_count_tl { \l__enumext_the_counter_ \__enumext_level: _tl }
618     \tl_put_right:ce { \l__enumext_renew_the_count_ \__enumext_level: _tl }
619     {
620       \exp_not:N \renewcommand { \exp_not:N \l__enumext_ref_the_count_tl }
621       { \exp_not:N \l__enumext_ref_key_arg_tl }
622     }
623   }
624 }

```

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

```

625 \cs_new_protected:Npn \__enumext_standar_ref:
626 {
627   \tl_if_empty:cF { \l__enumext_renew_the_count_ \__enumext_level: _tl }
628   {
629     \tl_use:c { \l__enumext_renew_the_count_ \__enumext_level: _tl }
630   }
631 }

```

(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
632 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
633 {
634   \keys_define:nn { enumext / #1 }
635   {
636     label .code:n = {
637       \__enumext_label_style:cvn { \l__enumext_label_#2_tl }
638       { \l__enumext_counter_#2_tl } {##1}
639       \dim_set_eq:cN { \l__enumext_labelwidth_#2_dim }
640       \l__enumext_current_widest_dim
641     },
642     label .initial:n = #3,
643     label .value_required:n = true,
644     ref .code:n = \__enumext_starred_ref:n {##1},
645     ref .value_required:n = true,
646   }
647 }
648 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
649 \__enumext_tmp:nnn { keyans* } { viii } { \Alph*.}

```

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

`__enumext_starred_ref:n`
`__enumext_starred_ref:`

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

```

650 \cs_new_protected:Npn \__enumext_starred_ref:n #1
651 {
652   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
653   \int_compare:nNt { \l__enumext_level_h_int } = { 1 }
654   {
655     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
656     {
657       \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
658     }
659     {

```

```

660         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
661         \__enumext_regex_counter_style:
662         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
663         \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
664         {
665             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
666             { \exp_not:V \l__enumext_ref_key_arg_tl }
667         }
668     }
669 }
670 \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
671 {
672     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
673     {
674         \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
675     }
676     {
677         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
678         \__enumext_regex_counter_style:
679         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
680         \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
681         {
682             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
683             { \exp_not:V \l__enumext_ref_key_arg_tl }
684         }
685     }
686 }
687 }

```

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.

```

688 \cs_new_protected:Nn \__enumext_starred_ref:
689 {
690     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
691     {
692         \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
693         {
694             \tl_use:N \l__enumext_renew_the_count_vii_tl
695         }
696     }
697     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
698     {
699         \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
700         {
701             \tl_use:N \l__enumext_renew_the_count_viii_tl
702         }
703     }
704 }

```

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

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

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

```

\l__enumext_label_v_tl
\l__enumext_label_vi_tl
705 \keys_define:nn { enumext / keyans }
706 {
707     label .code:n = {
708         \__enumext_label_style:cvn { \l__enumext_label_v_tl }
709         { \l__enumext_counter_v_tl } {#1}
710         \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
711         \l__enumext_current_widest_dim
712         \__enumext_label_style:cvn { \l__enumext_label_vi_tl }
713         { \l__enumext_counter_vi_tl } {#1}
714         \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
715         \l__enumext_current_widest_dim
716     },
717     label .initial:n = \Alph*,
718     label .value_required:n = true,
719     ref .code:n = \__enumext_keyans_ref:n {#1},
720     ref .value_required:n = true,
721 }

```

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

`__enumext_keyans_ref:n`
`__enumext_keyans_ref:`

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

```

722 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
723 {
724   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
725   \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
726   {
727     \msg_error:nnn { enumext } { key-ref-empty } { keyans }
728   }
729   {
730     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
731     \__enumext_regex_counter_style:
732     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
733     \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
734     {
735       \exp_not:N \renewcommand { \exp_not:N \l__enumext_ref_the_count_tl }
736       { \exp_not:N \l__enumext_ref_key_arg_tl }
737     }
738   }
739 }

```

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

```

740 \cs_new_protected:Nn \__enumext_keyans_ref:
741 {
742   \tl_if_empty:NF \l__enumext_renew_the_count_v_tl
743   {
744     \tl_use:N \l__enumext_renew_the_count_v_tl
745   }
746 }

```

(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.

```

747 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
748 {
749   \__enumext_if_is_int:nTF { #3 }
750   {
751     \int_set:Nn #2 {#3}
752   }
753   {
754     \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
755     { \int_set:Nn #2 { \int_from_alph:n {#3} } }
756     \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
757     { \int_set:Nn #2 { \int_from_roman:n {#3} } }
758   }
759 }
760 \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.

```

761 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
762 {

```

```

763     \__enumext_if_is_int:nTF {#4}
764     {
765         \setcounter{enumX#1} { #4 }
766     }
767     {
768         \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
769         { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
770         \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
771         { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
772     }
773     \__enumext_label_width_by_box:cv
774     { l__enumext_labelwidth_#1_dim } { l__enumext_label_#1_tl }
775 }
776 \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.

```

777 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
778 {
779     \keys_define:nn { enumext / #1 }
780     {
781         start* .code:n = {
782             \__enumext_start_from:ccn
783             { l__enumext_label_#2_tl }
784             { l__enumext_start_#2_int } {##1}
785         },
786         start* .value_required:n = true,
787         start .code:n = {
788             \__enumext_start_from:cce
789             { l__enumext_label_#2_tl }
790             { l__enumext_start_#2_int } { \int_eval:n {##1} }
791         },
792         start .initial:n = 1,
793         start .value_required:n = true,
794         widest .code:n = {
795             \__enumext_widest_from:nccn {#2}
796             { l__enumext_label_#2_tl }
797             { l__enumext_labelwidth_#2_dim } {##1}
798         },
799         widest .value_required:n = true,
800     }
801 }
802 \clist_map_inline:Nn \__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.

```

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

```

```

822             itemsep = 0pt, parsep= 0pt,
823             topsep = 0pt, partopsep = 0pt,
824         },
825         noseprule .value_forbidden:n = true,
826     }
827 }

```

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

```

828 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
829 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
830 { 4.0pt plus 2.0pt minus 1.0pt }
831 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
832 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
833 { 2.0pt plus 1.0pt minus 1.0pt }
834 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
835 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
836 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
837 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
838 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
839 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
840 { 2.0pt plus 1.0pt minus 1.0pt }
841 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
842 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
843 { 4.0pt plus 2.0pt minus 1.0pt }
844 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
845 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
846 { 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:
847 \cs_set_protected:Npn \__enumext_tmp:n #1
848 {
849     \keys_define:nn { enumext / #1 }
850     {
851         base-fix .bool_set:N = \l__enumext_base_line_fix_bool,
852         base-fix .initial:n = false,
853         base-fix .value_forbidden:n = true,
854     }
855 }
856 \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)

```

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

```

```

876         \keys_set:nn { enumext / enumext* }
877         {
878             topsep = 0pt, above = 0pt, above* = 0pt,
879         }
880     }
881     \bool_set_false:N \__enumext_base_line_fix_bool
882 }

```

🔗 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.

```

883 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
884 {
885     \keys_define:nn { enumext / #1 }
886     {
887         itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
888         itemindent .value_required:n = true,
889         rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
890         rightmargin .value_required:n = true,
891         listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
892         listparindent .value_required:n = true,
893         list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
894         list-offset .value_required:n = true,
895         list-indent .code:n =
896             \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
897             \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
898         list-indent .value_required:n = true,
899     }
900 }
901 \clist_map_inline:nn
902 {
903     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
904 }
905 { \__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.

```

906 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
907 {
908     \keys_define:nn { enumext / #1 }
909     {
910         itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
911         itemindent .value_required:n = true,
912         rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
913         rightmargin .value_required:n = true,
914         listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
915         listparindent .value_required:n = true,
916         list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
917         list-offset .value_required:n = true,
918         list-indent .meta:n = { list-offset = ##1 },
919         list-indent .value_required:n = true,
920     }
921 }
922 \clist_map_inline:nn
923 {
924     {enumext*}{vii}, {keyans*}{viii}
925 }
926 { \__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 `0pt`. Here I will need to place `\mode_leave_vertical:` and the plain TeX macro `\ignorespaces` to avoid unwanted extra space when using the `itemindent` key.

```

927 \cs_set_protected:Nn \__enumext_fake_item:
928 {

```



```

929 \dim_compare:nNtT
930 { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
931 >
932 { \c_zero_dim }
933 {
934   \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
935   {
936     \exp_not:N \mode_leave_vertical:
937     \exp_not:n { \skip_horizontal:n }
938     { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
939     \ignorespaces
940   }
941 }
942 }
943 \cs_set_protected:Nn \__enumext_keyans_fake_item:
944 {
945   \dim_compare:nNtT
946   { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
947   {
948     \tl_set:Nc \l__enumext_fake_item_indent_v_tl
949     {
950       \exp_not:N \mode_leave_vertical:
951       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
952     }
953   }
954 }
955 \cs_set_protected:Nn \__enumext_fake_item_vii:
956 {
957   \dim_compare:nNtT
958   { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
959   {
960     \tl_set:Nc \l__enumext_fake_item_indent_vii_tl
961     {
962       \exp_not:N \mode_leave_vertical:
963       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
964     }
965   }
966 }
967 \cs_set_protected:Nn \__enumext_fake_item_viii:
968 {
969   \dim_compare:nNtT
970   { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
971   {
972     \tl_set:Nc \l__enumext_fake_item_indent_viii_tl
973     {
974       \exp_not:N \mode_leave_vertical:
975       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
976     }
977   }
978 }

```

(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.

```

979 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
980 {
981   \keys_define:nn { enumext / #1 }
982   {
983     show-length .bool_set:c = { \l__enumext_show_length_#2_bool },
984     show-length .initial:n = false,
985   }
986 }
987 \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

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

```

before
before*
after
first
988 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
989 {
990   \keys_define:nn { enumext / #1 }
991   {
992     before .tl_set:c = { l__enumext_before_no_starred_key_#2_tl },
993     before .value_required:n = true,
994     before* .tl_set:c = { l__enumext_before_starred_key_#2_tl },
995     before* .value_required:n = true,
996     after .tl_set:c = { l__enumext_after_stop_list_#2_tl },
997     after .value_required:n = true,
998     first .tl_set:c = { l__enumext_after_list_args_#2_tl },
999     first .value_required:n = true,
1000   }
1001 }
1002 \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⟩}`.

```

1003 \cs_new_protected:Nn \__enumext_before_args_exec:
1004 {
1005   \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
1006 }

```

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⟩}`

```

1007 \cs_new_protected:Nn \__enumext_before_keys_exec:
1008 {
1009   \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
1010 }

```

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

```

1011 \cs_new_protected:Nn \__enumext_after_stop_list:
1012 {
1013   \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
1014 }

```

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.`

```

1015 \cs_new_protected:Nn \__enumext_after_args_exec:
1016 {
1017   \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
1018 }

```

(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:
1019 \cs_new_protected:Nn \__enumext_before_args_exec_v:
1020 {
1021   \tl_use:N \l__enumext_before_starred_key_v_tl
1022 }
1023 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
1024 {
1025   \tl_use:N \l__enumext_before_no_starred_key_v_tl
1026 }
1027 \cs_new_protected:Nn \__enumext_after_stop_list_v:
1028 {
1029   \tl_use:N \l__enumext_after_stop_list_v_tl
1030 }
1031 \cs_new_protected:Nn \__enumext_after_args_exec_v:
1032 {

```

```

1033     \tl_use:N \l__enumext_after_list_args_v_tl
1034   }

```

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

```

\__enumext_before_keys_exec_vii:
\__enumext_after_stop_list_vii:
\__enumext_after_args_exec_vii:
1035 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
1036 {
1037   \tl_use:N \l__enumext_before_starred_key_vii_tl
1038 }
1039 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
1040 {
1041   \tl_use:N \l__enumext_before_starred_key_viii_tl
1042 }
1043 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
1044 {
1045   \tl_use:N \l__enumext_before_no_starred_key_vii_tl
1046 }
1047 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
1048 {
1049   \tl_use:N \l__enumext_before_no_starred_key_viii_tl
1050 }
1051 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
1052 {
1053   \tl_use:N \l__enumext_after_stop_list_vii_tl
1054 }
1055 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
1056 {
1057   \tl_use:N \l__enumext_after_stop_list_viii_tl
1058 }
1059 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
1060 {
1061   \tl_use:N \l__enumext_after_list_args_vii_tl
1062 }
1063 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
1064 {
1065   \tl_use:N \l__enumext_after_list_args_viii_tl
1066 }

```

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

12.20 Setting keys for `multicols` and `minipage`

`mini-env` 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.

```

1067 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1068 {
1069   \keys_define:nn { enumext / #1 }
1070   {
1071     mini-env .dim_set:c = { \l__enumext_minipage_right_#2_dim },
1072     mini-env .value_required:n = true,
1073     mini-sep .dim_set:c = { \l__enumext_minipage_hsep_#2_dim },
1074     mini-sep .initial:n = 0.3333em,
1075     mini-sep .value_required:n = true,
1076     columns-sep .dim_set:c = { \l__enumext_columns_sep_#2_dim },
1077     columns-sep .value_required:n = true,
1078     columns .int_set:c = { \l__enumext_columns_#2_int },
1079     columns .initial:n = 1,
1080     columns .value_required:n = true,
1081   }
1082 }
1083 \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.

```

1084 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1085 {
1086   \keys_define:nn { enumext / #1 }

```

```

1087 {
1088   mini-right .tl_gset:c = { g__enumext_miniright_code_#2_tl },
1089   mini-right .value_required:n = true,
1090   mini-right* .code:n = {
1091     \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
1092     \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1093   },
1094   mini-right* .value_required:n = true,
1095 }
1096 }
1097 \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.

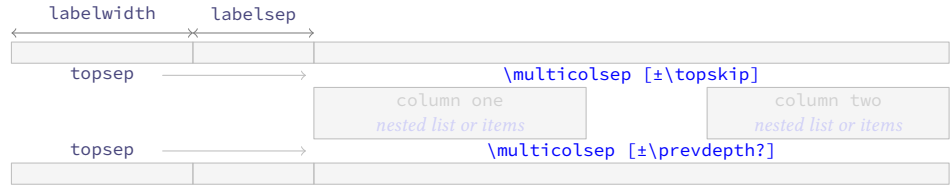


Figure 7: Representation of the vertical space in `multicol` for a nested level.

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 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 T_EX 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*.

```

1098 \cs_new_protected:Nn \__enumext_multi_set_vskip:
1099 {
1100   \skip_set:cn { l__enumext_multicol_above_ \__enumext_level: _skip }
1101   {
1102     \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1103   }
1104   \skip_set:cn { l__enumext_multicol_below_ \__enumext_level: _skip }
1105   {
1106     \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1107   }
1108   \__enumext_add_pre_parsep:
1109 }

```

(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*.

```

1110 \cs_new_protected:Nn \__enumext_add_pre_parsep:
1111 {
1112   \int_case:nn { \l__enumext_level_int }
1113   {
1114     { 2 }{
1115       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1116       {
1117         \skip_add:Nn \l__enumext_multicol_above_ii_skip { \l__enumext_parsep_i_skip }
1118       }
1119     }
1120   }

```

```

1119         }
1120     { 3 }{
1121         \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1122         {
1123             \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_ii_skip
1124         }
1125     }
1126     { 4 }{
1127         \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1128         {
1129             \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip
1130         }
1131     }
1132 }
1133 }

```

(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*.

```

1134 \cs_new_protected:Nn \__enumext_multi_addvspace:
1135 {
1136     \__enumext_multi_set_vskip:
1137     \mode_if_vertical:T
1138     {
1139         \skip_add:cn { \l__enumext_multicols_above_ \l__enumext_level: _skip }
1140         {
1141             \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1142         }
1143         \skip_add:cn { \l__enumext_multicols_below_ \l__enumext_level: _skip }
1144         {
1145             \skip_use:c { \l__enumext_partopsep_ \l__enumext_level: _skip }
1146         }
1147     }
1148     %%\__enumext_unskip_unkern:
1149     \par\nopagebreak
1150     \addvspace{ \skip_use:c { \l__enumext_multicols_above_ \l__enumext_level: _skip } }
1151 }

```

(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:`

```

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

```

```

1177 \__enumext_unskip_unkern:
1178 \par\nopagebreak
1179 \addvspace{ \l__enumext_multicols_above_v_skip }
1180 }

```

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

```

1181 \cs_new_protected:Nn \__enumext_minipage_set_skip:
1182 {
1183   \skip_set:Nn \l__enumext_minipage_right_skip
1184   {
1185     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1186   }
1187   \mode_if_vertical:T
1188   {
1189     \skip_add:Nn \l__enumext_minipage_right_skip
1190     {
1191       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1192     }
1193   }
1194   \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip

```

We will adjust the values `\l__enumext_multicols_above_X_skip` and `\l__enumext_multicols_below_X_skip` and call the function `__enumext_pre_itemsep_skip:`.

```

1195   \skip_set_eq:cN
1196   { \l__enumext_multicols_above_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1197   \skip_set_eq:cN
1198   { \l__enumext_multicols_below_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1199   \__enumext_pre_itemsep_skip:

```

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

```

1200   \int_compare:nNnT
1201   { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1202   {
1203     \skip_zero:N \topskip
1204     \skip_set_eq:Nc \multicolsep { \l__enumext_multicols_above_ \__enumext_level: _skip }
1205   }
1206 }

```


The function `__enumext_minipage_add_space:` will apply the spaces on the “left side” using `\addvspace` “above” the `__enumext_minipage` 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*.

```

1207 \cs_new_protected:Nn \__enumext_minipage_add_space:
1208 {
1209   \__enumext_minipage_set_skip:
1210   \__enumext_unskip_unkern:
1211   \mode_if_vertical:TF
1212   {
1213     \nopagebreak\nointerlineskip
1214   }
1215   {
1216     \par\nopagebreak\nointerlineskip
1217     \skip_zero:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1218   }
1219   \int_compare:nNnTF
1220   { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1221   {
1222     \addvspace{ 0.445\box_ht:N \strutbox }
1223   }
1224   {
1225     \addvspace{ 0.250\box_ht:N \strutbox }
1226   }
1227 }

```

(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.

```

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

```

```

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

```

```

1337         }
1338         \skip_sub:Nn
1339         \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1340         \skip_sub:Nn
1341         \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1342         \skip_add:Nn
1343         \l__enumext_minipage_after_skip
1344         { 0.150\box_ht:N \strutbox + \l_tmpa_skip }
1345         \skip_add:Nn
1346         \l__enumext_multicols_below_iv_skip
1347         { 0.350\box_ht:N \strutbox + \l_tmpa_skip }
1348     }
1349 }
1350 }
1351 }
1352 }

```

(End of definition for `__enumext_pre_itemsep_skip:`)

12.22.2 Adjustment of vertical spaces for minipage in keyans

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`.

```

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

```

```

1401 \skip_if_eq:nnTF
1402 { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1403 {
1404   \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1405   \skip_set:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1406 }
1407 {
1408   \dim_compare:nNnT
1409     { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1410     {
1411       \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1412       \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1413       \skip_add:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1414       \skip_add:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1415     }
1416   \dim_compare:nNnT
1417     { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1418     {
1419       \skip_set:Nn \l_tmpa_skip
1420       {
1421         \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1422       }
1423       \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1424       \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1425       \skip_add:Nn \l__enumext_minipage_after_skip
1426         { 0.150\box_ht:N \strutbox + \l_tmpa_skip }
1427       \skip_add:Nn \l__enumext_multicols_below_v_skip
1428         { 0.350\box_ht:N \strutbox + \l_tmpa_skip }
1429     }
1430   }
1431 }

```

(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*`.

```

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

```

```

1464     }
1465     \skip_set:Nn \l__enumext_minipage_right_skip
1466     {
1467         \l__enumext_partopsep_viii_skip
1468     }
1469     \skip_set:Nn \l__enumext_minipage_after_skip
1470     {
1471         1.6\box_dp:N \strutbox
1472     }
1473 }
1474 {
1475     \skip_set:Nn \l__enumext_minipage_left_skip
1476     {
1477         0.5875\box_dp:N \strutbox
1478     }
1479     \skip_set:Nn \l__enumext_minipage_right_skip
1480     {
1481         \l__enumext_topsep_viii_skip
1482     }
1483     \skip_set:Nn \l__enumext_minipage_after_skip
1484     {
1485         0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1486     }
1487 }
1488 }

```

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

```

1489 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1490 {
1491     \__enumext_mini_set_vskip_vii:
1492     \par\nopagebreak
1493     \addvspace { \l__enumext_minipage_left_skip }
1494 }
1495 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1496 {
1497     \__enumext_mini_set_vskip_viii:
1498     \par\nopagebreak
1499     \addvspace { \l__enumext_minipage_left_skip }
1500 }

```

(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.

```

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

```

```

1513     }
1514     % starred env
1515     \bool_if:NT \l__enumext_starred_bool
1516     {
1517         \msg_error:nnn { enumext } { wrong-miniright-starred }
1518     }
1519     \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
1520     {
1521         \__enumext_keyans_mini_right_cmd:n {#1}
1522     }
1523     { \__enumext_mini_right_cmd:n {#1} }
1524 }

```

(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.

```

1525 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1526 {
1527     \dim_compare:nNnTF
1528     { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1529     {
1530         \__enumext_multicols_stop:
1531         \int_compare:nNnT
1532         { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } = { 1 }
1533         {
1534             \par\addvspace{ \l__enumext_minipage_after_skip }
1535         }
1536         \end__enumext_mini_page
1537         \hfill
1538         \__enumext_mini_page{ \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } }
1539         \par\nointerlineskip
1540         \addvspace { \l__enumext_minipage_right_skip }
1541         \bool_if:nF {#1}
1542         {
1543             \centering
1544         }
1545         \int_gzero:N \g__enumext_minipage_stat_int
1546     }
1547     { \msg_error:nnn { enumext } { wrong-miniright-use } }
1548     % paranoia
1549     \RenewDocumentCommand \miniright { s }
1550     {
1551         \msg_error:nn { enumext } { many-miniright-used }
1552     }
1553 }

```

(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.

```

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

```



```

1569         {
1570             \centering
1571         }
1572         \int_gzero:N \g__enumext_minipage_stat_int
1573     }
1574     { \msg_error:nnn { enumext } { wrong-miniright-use } }
1575 % paranoia
1576 \RenewDocumentCommand \miniright { s }
1577 {
1578     \msg_error:nn { enumext } { many-miniright-used }
1579 }
1580 }

```

(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*  Define above, above*, below and below* keys for enumext and keyans environments.
below*  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
below*  {
1581 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1582 {
1583     \keys_define:nn { enumext / #1 }
1584     {
1585         above .skip_set:c = { l__enumext_vspace_above_#2_skip },
1586         above .value_required:n = true,
1587         above* .code:n      = \bool_set_true:c { l__enumext_vspace_a_star_#2_bool }
1588             \keys_set:nn { enumext / #1 } { above = {##1} },
1589         above* .value_required:n = true,
1590         below .skip_set:c = { l__enumext_vspace_below_#2_skip },
1591         below .value_required:n = true,
1592         below* .code:n      = \bool_set_true:c { l__enumext_vspace_b_star_#2_bool }
1593             \keys_set:nn { enumext / #1 } { below = {##1} },
1594         below* .value_required:n = true,
1595     }
1596 }
1597 \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.

```

1598 \cs_new_protected:Nn \__enumext_vspace_above:
1599 {
1600     \skip_if_eq:nnF
1601     { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1602     {
1603         \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
1604         {
1605             \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1606         }
1607         {
1608             \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1609         }
1610     }
1611 }

```

(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.

```

1612 \cs_new_protected:Nn \__enumext_vspace_below:
1613 {
1614     \skip_if_eq:nnF
1615     { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1616     {

```

```

1617         \bool_if:cTF { \__enumext_vspace_b_star_ \__enumext_level: _bool }
1618         {
1619             \vspace*{ \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1620         }
1621         {
1622             \vspace { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1623         }
1624     }
1625 }

```

(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.

```

1626 \cs_new_protected:Nn \__enumext_vspace_above_v:
1627 {
1628     \skip_if_eq:nnF { \__enumext_vspace_above_v_skip } { \c_zero_skip }
1629     {
1630         \bool_if:NTF \__enumext_vspace_a_star_v_bool
1631         {
1632             \vspace*{ \__enumext_vspace_above_v_skip }
1633         }
1634         { \vspace { \__enumext_vspace_above_v_skip } }
1635     }
1636 }

```

(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.

```

1637 \cs_new_protected:Nn \__enumext_vspace_below_v:
1638 {
1639     \skip_if_eq:nnF { \__enumext_vspace_below_v_skip } { \c_zero_skip }
1640     {
1641         \bool_if:NTF \__enumext_vspace_b_star_v_bool
1642         {
1643             \vspace*{ \__enumext_vspace_below_v_skip }
1644         }
1645         { \vspace { \__enumext_vspace_below_v_skip } }
1646     }
1647 }

```

(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:

```

1648 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1649 {
1650     \skip_if_eq:nnF { \__enumext_vspace_above_vii_skip } { \c_zero_skip }
1651     {
1652         \bool_if:NTF \__enumext_vspace_a_star_vii_bool
1653         {
1654             \vspace*{ \__enumext_vspace_above_vii_skip }
1655         }
1656         { \vspace { \__enumext_vspace_above_vii_skip } }
1657     }
1658 }
1659 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1660 {
1661     \skip_if_eq:nnF { \__enumext_vspace_above_viii_skip } { \c_zero_skip }
1662     {
1663         \bool_if:NTF \__enumext_vspace_a_star_viii_bool
1664         {
1665             \vspace*{ \__enumext_vspace_above_viii_skip }
1666         }
1667         { \vspace { \__enumext_vspace_above_viii_skip } }
1668     }
1669 }

```

(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.

```

1670 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1671 {
1672   \skip_if_eq:nnF { \__enumext_vspace_below_vii_skip } { \c_zero_skip }
1673   {
1674     \bool_if:NTF \__enumext_vspace_b_star_vii_bool
1675     {
1676       \vspace*{ \__enumext_vspace_below_vii_skip }
1677     }
1678     { \vspace { \__enumext_vspace_below_vii_skip } }
1679   }
1680 }
1681 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1682 {
1683   \skip_if_eq:nnF { \__enumext_vspace_below_viii_skip } { \c_zero_skip }
1684   {
1685     \bool_if:NTF \__enumext_vspace_b_star_viii_bool
1686     {
1687       \vspace*{ \__enumext_vspace_below_viii_skip }
1688     }
1689     { \vspace { \__enumext_vspace_below_viii_skip } }
1690   }
1691 }

```

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

12.24 Setting series, resume and resume* keys

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

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

```

series
resume
resume*
1692 \cs_set_protected:Npn \__enumext_tmp:n #1
1693 {
1694   \keys_define:nn { enumext / #1 }
1695   {
1696     series .str_set:N = \__enumext_series_str,
1697     series .value_required:n = true,
1698     resume .code:n = \__enumext_resume_series:n {##1},
1699     resume* .code:n = \__enumext_resume_starred:,
1700     resume* .value_forbidden:n = true,
1701   }
1702 }
1703 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

12.24.1 Internal functions for series key

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

```

1704 \cs_new:Npn \__enumext_filter_series:n #1
1705 {
1706   \use:e
1707   {
1708     \keyval_parse:NNn
1709     \__enumext_filter_series_key:n
1710     \__enumext_filter_series_pair:nn {#1}
1711   }
1712 }

```

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

```

1713 \cs_new:Npn \__enumext_filter_series_key:n #1
1714 {
1715   \str_case:nnF {#1}
1716   {
1717     { resume } {} { resume* } {} { base-fix } {}

```

```

1718     }
1719     { , { \exp_not:n {#1} } }
1720 }

```

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.

```

1721 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1722 {
1723   \str_case:nnF {#1}
1724   {
1725     { series } {} { resume } {} { start } {}
1726     { start* } {} { save-ans } {} { save-key } {}
1727   }
1728   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1729 }

```

(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).

```

1730 \cs_new_protected:Npn \__enumext_parse_series:n #1
1731 {
1732   \str_if_empty:NTF \l__enumext_series_str
1733   {
1734     \bool_if:NF \l__enumext_resume_active_bool
1735     {
1736       \__enumext_resume_last:n {#1}
1737     }
1738   }
1739   {
1740     \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str _tl }
1741     \tl_gset:ce { g__enumext_series_ \l__enumext_series_str _tl }
1742     { \__enumext_filter_series:n {#1} }
1743     \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str _int }
1744     {
1745       \int_new:c { g__enumext_series_ \l__enumext_series_str _int }
1746     }
1747   }
1748 }

```

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.

```

1749 \cs_new_protected:Npn \__enumext_resume_last:n #1
1750 {
1751   \bool_if:NT \l__enumext_standar_first_bool
1752   {
1753     \tl_gclear:N \g__enumext_standar_series_tl
1754     \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1755   }
1756   \bool_if:NT \l__enumext_starred_first_bool
1757   {
1758     \tl_gclear:N \g__enumext_starred_series_tl
1759     \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1760   }
1761 }

```

(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).

```

1762 \cs_new_protected:Nn \__enumext_resume_save_counter:
1763 {
1764   \bool_if:NT \g__enumext_standar_bool
1765   {
1766     \tl_if_empty:NF \l__enumext_series_str
1767     {
1768       \int_gset_eq:cN
1769       { g__enumext_series_ \l__enumext_series_str_int } \value{enumXi}
1770     }
1771     \tl_if_empty:NTF \l__enumext_resume_name_tl
1772     {
1773       \str_if_empty:NT \l__enumext_series_str
1774       {
1775         \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1776       }
1777     }
1778     {
1779       \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl_int }
1780       {
1781         \int_gset_eq:cN
1782         { g__enumext_series_ \l__enumext_resume_name_tl_int } \value{enumXi}
1783       }
1784     }
1785     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl_int }
1786     {
1787       \int_gset_eq:cN
1788       { g__enumext_resume_ \l__enumext_store_name_tl_int } \value{enumXi}
1789     }
1790   }
1791   \bool_if:NT \g__enumext_starred_bool
1792   {
1793     \tl_if_empty:NF \l__enumext_series_str
1794     {
1795       \int_gset_eq:cN
1796       { g__enumext_series_ \l__enumext_series_str_int } \value{enumXvii}
1797     }
1798     \tl_if_empty:NTF \l__enumext_resume_name_tl
1799     {
1800       \str_if_empty:NT \l__enumext_series_str
1801       {
1802         \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1803       }
1804     }
1805     {
1806       \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl_int }
1807       {
1808         \int_gset_eq:cN
1809         { g__enumext_series_ \l__enumext_resume_name_tl_int } \value{enumXvii}
1810       }
1811     }
1812     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl_int }
1813     {
1814       \int_gset_eq:cN
1815       { g__enumext_resume_ \l__enumext_store_name_tl_int } \value{enumXvii}
1816     }
1817   }
1818 }

```

(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.

```

1819 \cs_new_protected:Npn \__enumext_resume_series:n #1
1820 {
1821   \tl_if_empty:nTF {#1}
1822   {
1823     \__enumext_resume_counter:n { }
1824   }
1825   {
1826     \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1827     {
1828       \__enumext_resume_counter:n {#1}
1829       \bool_if:NT \g__enumext_standar_bool
1830       {
1831         \keys_set:nv { enumext / level-1 }
1832         { g__enumext_series_ \tl_to_str:n {#1} _tl }
1833       }
1834       \bool_if:NT \g__enumext_starred_bool
1835       {
1836         \keys_set:nv { enumext / enumext* }
1837         { g__enumext_series_ \tl_to_str:n {#1} _tl }
1838       }
1839     }
1840     {
1841       \bool_if:NT \g__enumext_standar_bool
1842       {
1843         \msg_error:nnn { enumext } { unknown-series } {#1}
1844       }
1845       \bool_if:NT \g__enumext_starred_bool
1846       {
1847         \msg_error:nnn { enumext } { unknown-series } {#1}
1848       }
1849     }
1850   }
1851 }

```

(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`.

```

1852 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1853 {
1854   \bool_set_true:N \l__enumext_resume_active_bool
1855   \tl_set:Nn \l__enumext_resume_name_tl {#1}
1856   \tl_if_empty:NTF \l__enumext_resume_name_tl
1857   {
1858     \__enumext_resume_counter:
1859   }
1860   {
1861     \__enumext_resume_counter_series:
1862   }
1863   \__enumext_resume_counter_save_ans:
1864 }

```

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.

```

1865 \cs_new_protected:Nn \__enumext_resume_counter:
1866 {
1867   \bool_if:NT \g__enumext_standar_bool

```

```

1868     {
1869         \int_gincr:N \g__enumext_resume_int
1870         \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1871     }
1872     \bool_if:NT \g__enumext_starred_bool
1873     {
1874         \int_gincr:N \g__enumext_resume_vii_int
1875         \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1876     }
1877 }

```

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.

```

1878 \cs_new_protected:Nn \__enumext_resume_counter_series:
1879 {
1880     \bool_if:NT \g__enumext_standar_bool
1881     {
1882         \int_set:Nn \l__enumext_start_i_int
1883         {
1884             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1885         }
1886     }
1887     \bool_if:NT \g__enumext_starred_bool
1888     {
1889         \int_set:Nn \l__enumext_start_vii_int
1890         {
1891             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1892         }
1893     }
1894 }

```

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.

```

1895 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1896 {
1897     \bool_lazy_and:nnT
1898     { \bool_if_p:N \l__enumext_standar_first_bool }
1899     { \bool_if_p:N \l__enumext_store_active_bool }
1900     {
1901         \int_set:Nn \l__enumext_start_i_int
1902         {
1903             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1904         }
1905     }
1906     \bool_lazy_and:nnT
1907     { \bool_if_p:N \l__enumext_starred_first_bool }
1908     { \bool_if_p:N \l__enumext_store_active_bool }
1909     {
1910         \int_set:Nn \l__enumext_start_vii_int
1911         {
1912             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1913         }
1914     }
1915 }

```

(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.

```

1916 \cs_new_protected:Nn \__enumext_resume_starred:
1917 {
1918     \bool_if:NT \g__enumext_standar_bool
1919     {
1920         \tl_if_empty:NF \g__enumext_standar_series_tl
1921         {
1922             \__enumext_resume_counter:n { }

```



```

1923         \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1924     }
1925 }
1926 \bool_if:NT \g__enumext_starred_bool
1927 {
1928     \tl_if_empty:NF \g__enumext_starred_series_tl
1929     {
1930         \__enumext_resume_counter:n { }
1931         \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1932     }
1933 }
1934 }

```

(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*`.

```

1935 \cs_set_protected:Npn \__enumext_tmp:n #1
1936 {
1937     \keys_define:nn { enumext / #1 }
1938     {
1939         save-ans .code:n = \__enumext_storing_set:n {##1},
1940         save-ans .value_required:n = true,
1941     }
1942 }
1943 \clist_map_inline:nn { level-1, enumext* } { { \__enumext_tmp:n {#1} } }

```

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

12.25.2 Internal functions for save-ans key

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

```

1944 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
1945 {
1946     \msg_term:nnVV { enumext } { save-ans-log }
1947     \g__enumext_envir_name_tl \l__enumext_store_name_tl
1948 }
1949 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
1950 {
1951     \msg_term:nnVV { enumext } { save-ans-log-hook }
1952     \g__enumext_envir_name_tl \g__enumext_store_name_tl
1953 }

```

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

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

```

1954 \cs_new_protected:Npn \__enumext_storing_set:n #1
1955 {
1956     \tl_set:Ne \l__enumext_store_name_tl {#1}
1957     \tl_if_empty:NTF \l__enumext_store_name_tl
1958     {
1959         \bool_lazy_or:nnT
1960         { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1961         {
1962             \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
1963         }
1964     }
1965     {
1966         \bool_lazy_or:nnT

```

```

1967         { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1968     {
1969         __enumext_start_save_ans_msg:
1970         __enumext_storing_exec:
1971     }
1972 }
1973 }

```

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*`.

```

1974 \cs_new_protected:Nn \__enumext_storing_exec:
1975 {
1976     \bool_set_true:N \l__enumext_store_active_bool
1977     \bool_set_true:N \l__enumext_check_answers_bool
1978     \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1979     \__enumext_anskey_env_make:V \l__enumext_store_name_tl
1980     \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1981     {
1982         \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
1983         \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1984     }
1985     \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1986     {
1987         \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
1988         \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1989     }
1990     \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1991     {
1992         \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
1993         \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1994     }
1995 }

```

(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 1996 \cs_set_protected:Npn \__enumext_tmp:n #1
no-store 1997 {
1998   \keys_define:nn { enumext / #1 }
1999   {
2000     check-ans .bool_set:N = \__enumext_check_ans_key_bool,
2001     check-ans .initial:n = false,
2002     check-ans .value_required:n = true,
2003     no-store .code:n = {
2004       \bool_set_false:N \__enumext_check_answers_bool
2005       \bool_set_false:N \__enumext_check_ans_key_bool
2006     },
2007     no-store .value_forbidden:n = true,
2008   }
2009 }
2010 \clist_map_inline:nn
2011 {
2012   level-1, level-2, level-3, level-4, enumext*
2013 }
2014 { \__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.

```

2015 \cs_new_protected:Nn \__enumext_check_ans_active:
2016 {
2017   \tl_if_empty:NF \__enumext_store_name_tl
2018   {
2019     \bool_if:NT \__enumext_check_answers_bool
2020     {
2021       \__enumext_check_ans_level:
2022     }
2023   }
2024 }
```

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*”.

```

2025 \cs_new_protected:Nn \__enumext_check_ans_level:
2026 {
2027   \int_case:nn { \__enumext_level_int }
2028   {
2029     { 1 }{
2030       \bool_lazy_all:nT
2031       {
2032         { \bool_if_p:N \g__enumext_starred_bool }
2033         { \int_compare_p:nNn { \__enumext_level_h_int } = { 1 } }
2034       }
2035       {
2036         \int_gdecr:N \g__enumext_item_number_int
2037         \bool_set_false:N \__enumext_item_number_bool
2038       }
2039     }
2040     { 2 }{
2041       \int_gdecr:N \g__enumext_item_number_int
2042       \bool_set_false:N \__enumext_item_number_bool
2043     }
2044     { 3 }{
2045       \int_gdecr:N \g__enumext_item_number_int
2046       \bool_set_false:N \__enumext_item_number_bool
2047     }
2048     { 4 }{
2049       \int_gdecr:N \g__enumext_item_number_int
2050       \bool_set_false:N \__enumext_item_number_bool

```

```

2051         }
2052     }

```

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.

```

2053     \int_case:nn { \l__enumext_level_h_int }
2054     {
2055         { 1 }{
2056             \bool_lazy_all:nT
2057             {
2058                 { \bool_if_p:N \g__enumext_standar_bool }
2059                 { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
2060             }
2061             {
2062                 \int_gdecr:N \g__enumext_item_number_int
2063                 \bool_set_false:N \l__enumext_item_number_bool
2064             }
2065         }
2066     }
2067 }

```

(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.

```

2068 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
2069 {
2070     \bool_lazy_and:nnT
2071     { \bool_if_p:N \l__enumext_check_ans_key_bool }
2072     { \bool_if_p:N \g__enumext_standar_bool }
2073     {
2074         \bool_gset_true:N \g__enumext_check_ans_key_bool
2075     }
2076     \bool_lazy_and:nnT
2077     { \bool_if_p:N \l__enumext_check_ans_key_bool }
2078     { \bool_if_p:N \g__enumext_starred_bool }
2079     {
2080         \bool_gset_true:N \g__enumext_check_ans_key_bool
2081     }
2082 }

```

(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:`.

```

2083 \cs_new_protected:Nn \__enumext_item_answer_diff:
2084 {
2085     \int_gset:Nn \g__enumext_item_answer_diff_int
2086     {
2087         \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
2088     }
2089 }

```

(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:`.

```

2090 \cs_new_protected:Nn \__enumext_check_ans_show:
2091 {
2092     \int_case:nn { \g__enumext_item_answer_diff_int }
2093     {
2094         { -1 }{ \__enumext_check_ans_msg_less: }
2095         { 0 }{ \__enumext_check_ans_msg_same_ok: }
2096         { 1 }{ \__enumext_check_ans_msg_greater: }
2097     }
2098 }

```

```

2099 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
2100 {
2101   \msg_warning:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2102   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2103 }
2104 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
2105 {
2106   \msg_term:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2107   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2108 }
2109 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
2110 {
2111   \msg_warning:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2112   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2113 }

```

(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:`.

```

2114 \cs_new_protected:Nn \__enumext_check_ans_log:
2115 {
2116   \int_case:nn { \g__enumext_item_answer_diff_int }
2117   {
2118     { -1 } { \__enumext_check_ans_log_msg_less: }
2119     { 0 } { \__enumext_check_ans_log_msg_same_ok: }
2120     { 1 } { \__enumext_check_ans_log_msg_greater: }
2121   }
2122 }
2123 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
2124 {
2125   \msg_log:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2126   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2127 }
2128 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
2129 {
2130   \msg_log:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2131   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2132 }
2133 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
2134 {
2135   \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2136   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2137 }

```

(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.

```

2138 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
2139 {
2140   \int_compare:nNnT
2141   { \g__enumext_check_starred_cmd_int } = { 0 }
2142   {
2143     \msg_warning:nnnV
2144     { enumext } { missing-starred } { #1 } \l__enumext_check_start_line_env_tl
2145   }
2146   \int_compare:nNnT
2147   { \g__enumext_check_starred_cmd_int } > { 1 }
2148   {
2149     \msg_warning:nnnV
2150     { enumext } { many-starred } { #1 } \l__enumext_check_start_line_env_tl
2151   }
2152   \int_gzero:N \g__enumext_check_starred_cmd_int
2153   \tl_clear:N \l__enumext_check_start_line_env_tl
2154 }

```

(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*`.

```

2155 \cs_set_protected:Npn \__enumext_tmp:n #1
2156 {
2157   \keys_define:nn { enumext / #1 }
2158   {
2159     wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
2160     wrap-ans .initial:n =
2161       {
2162         \fbox{\parbox[t]{\dimeval{\itemwidth -2\fboxsep -2\fboxrule}}{##1}}
2163       },
2164     wrap-ans .value_required:n = true,
2165     wrap-opt .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
2166     wrap-opt .initial:n = [{##1}],
2167     wrap-opt .value_required:n = true,
2168     save-sep .tl_set:N = \__enumext_store_keyans_item_opt_sep_tl,
2169     save-sep .initial:n = {, ~ },
2170     save-sep .value_required:n = true,
2171     mark-ans .tl_set:N = \__enumext_mark_answer_sym_tl,
2172     mark-ans .initial:n = \textasteriskcentered,
2173     mark-ans .value_required:n = true,
2174     mark-pos .choice:,
2175     mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
2176     mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
2177     mark-pos / unknown .code:n =
2178       \msg_error:nneee { enumext } { unknown-choice }
2179       { mark-pos } { left, ~ right } { \exp_not:n {##1} },
2180     mark-pos .initial:n = right,
2181     mark-pos .value_required:n = true,
2182     show-ans .bool_set:N = \__enumext_show_answer_bool,
2183     show-ans .initial:n = false,
2184     show-ans .value_required:n = true,
2185     show-pos .bool_set:N = \__enumext_show_position_bool,
2186     show-pos .initial:n = false,
2187     show-pos .value_required:n = true,
2188     mark-ref .tl_set:N = \__enumext_mark_ref_sym_tl,
2189     mark-ref .initial:n = \textasteriskcentered,
2190     mark-ref .value_required:n = true,
2191     save-ref .bool_set:N = \__enumext_store_ref_key_bool,
2192     save-ref .initial:n = false,
2193     save-ref .value_required:n = true,
2194   }
2195 }
2196 \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`.

```

2197 \cs_set_protected:Npn \__enumext_tmp:n #1
2198 {
2199   \keys_define:nn { enumext / #1 }
2200   {
2201     mark-pos .choice:,
2202     mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
2203     mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
2204     mark-pos .initial:n = right,
2205     mark-pos .value_required:n = true,
2206     show-ans .bool_set:N = \__enumext_show_answer_bool,
2207     show-ans .initial:n = false,
2208     show-ans .value_required:n = true,
2209     show-pos .bool_set:N = \__enumext_show_position_bool,
2210     show-pos .initial:n = false,
2211     show-pos .value_required:n = true,
2212   }
2213 }
2214 \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*.

```

2215 \cs_new_protected:Npn __enumext_store_active_keys:n #1
2216 {
2217   \bool_if:cF { __enumext_store_save_key_ __enumext_level: _bool }
2218   {
2219     \tl_clear:c { __enumext_save_key_ __enumext_level: _tl }
2220     \tl_set:ce
2221       { __enumext_store_save_key_ __enumext_level: _tl }
2222       { __enumext_filter_save_key:n {#1} }
2223   }
2224 }
2225 \cs_new_protected:Npn __enumext_store_active_keys_vii:n #1
2226 {
2227   \bool_if:NF \__enumext_store_save_key_vii_bool
2228   {
2229     \tl_clear:N \__enumext_store_save_key_vii_tl
2230     \tl_set:Ne \__enumext_store_save_key_vii_tl { __enumext_filter_save_key:n {#1} }
2231   }
2232 }

```

(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.

```

2233 \cs_set_protected:Npn __enumext_tmp:n #1
2234 {
2235   \keys_define:nn { enumext / enumext* }
2236   {
2237     save-key .code:n = __enumext_parse_save_key_vii:n {##1},
2238     save-key .value_required:n = true,
2239   }
2240   \keys_define:nn { enumext / #1 }
2241   {
2242     save-key .code:n = __enumext_parse_save_key:n {##1},
2243     save-key .value_required:n = true,
2244   }
2245 }
2246 \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*`.

```

2247 \cs_new_protected:Npn __enumext_parse_save_key:n #1
2248 {
2249   \bool_set_true:c { __enumext_store_save_key_ __enumext_level: _bool }
2250   \tl_clear:c { __enumext_save_key_ __enumext_level: _tl }
2251   \tl_set:ce
2252     { __enumext_store_save_key_ __enumext_level: _tl }
2253     { __enumext_filter_save_key:n {#1} }
2254 }

```



```

2255 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2256 {
2257   \bool_set_true:N \l__enumext_store_save_key_vii_bool
2258   \tl_clear:N \l__enumext_store_save_key_vii_tl
2259   \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2260 }

```

(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

```

```

2261 \cs_new:Npn \__enumext_filter_save_key:n #1
2262 {
2263   \use:e
2264   {
2265     \keyval_parse:NNn
2266     \__enumext_filter_save_key_key:n
2267     \__enumext_filter_save_key_pair:nn {#1}
2268   }
2269 }

```

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.

```

2270 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2271 {
2272   \str_case:nnF {#1}
2273   {
2274     { resume } {} { resume* } {} { no-store } {} { base-fix } {}
2275   }
2276   { , { \exp_not:n {#1} } }
2277 }

```

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.

```

2278 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2279 {
2280   \str_case:nnF {#1}
2281   {
2282     { series } {} { resume } {} { save-ans } {} { save-ref } {}
2283     { save-key } {} { check-ans } {} { show-ans } {} { show-pos } {}
2284     { wrap-ans } {} { mark-ans } {} { wrap-opt } {} { save-sep } {}
2285     { mark-ref } {} { mini-env } {} { mini-sep } {} { mini-right } {}
2286     { mini-right* } {}
2287   }
2288   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2289 }

```

(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.

```

2290 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2291 {
2292   \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2293   {
2294     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2295   }
2296   { #1 }
2297 }
2298 \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.

```

2299 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2300 {
2301   \seq_gput_right:cn { g__enumext_ \__enumext_store_name_tl_seq } { #1 }
2302 }
2303 \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.

```

2304 \cs_new_protected:Nn \__enumext_store_level_open:
2305 {
2306   \bool_if:NT \__enumext_check_answers_bool
2307   {
2308     \tl_if_empty:CTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2309     {
2310       \__enumext_store_addto_seq:n
2311       {
2312         \item \begin{enumext}
2313       }
2314     }
2315   {
2316     \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2317     {
2318       \item \begin{enumext} [
2319     }
2320     \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2321     {
2322       ]
2323     }
2324     \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2325   }
2326 }
2327 }
2328 \cs_new_protected:Nn \__enumext_store_level_close:
2329 {
2330   \bool_if:NT \__enumext_check_answers_bool
2331   {
2332     \__enumext_store_addto_seq:n { \end{enumext} }
2333   }
2334 }

```

(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.

```

2335 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2336 {
2337   \bool_if:NT \__enumext_check_answers_bool
2338   {
2339     \tl_if_empty:NTF l__enumext_store_save_key_vii_tl
2340     {
2341       \__enumext_store_addto_seq:n
2342       {
2343         \item \begin{enumext*}
2344       }
2345     }
2346   {
2347     \tl_put_left:Nn l__enumext_store_save_key_vii_tl
2348     {
2349       \item \begin{enumext*}[
2350     }
2351     \tl_put_right:Nn l__enumext_store_save_key_vii_tl

```

```

2352         {
2353         }
2354     }
2355     \__enumext_store_addto_seq:V \__enumext_store_save_key_vii_tl
2356 }
2357 }
2358 }
2359 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2360 {
2361     \bool_if:NT \__enumext_check_answers_bool
2362     {
2363         \__enumext_store_addto_seq:n { \end{enumext*} }
2364     }
2365 }

```

(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

```

2366 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2367 {
2368     \mode_leave_vertical:
2369     \skip_horizontal:n { -\dim_use:N #2 }
2370     \makebox[0pt][ r ]
2371     {
2372         \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2373         {
2374             \tl_use:N \l__enumext_mark_answer_sym_tl
2375         }
2376     }
2377     \skip_horizontal:n { \dim_use:N #2 }
2378 }
2379 \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.

```

2380 \cs_new_protected:Nn \__enumext_store_internal_ref:
2381 {
2382     \cs_set_protected:Npn \__enumext_tmp:n ##1
2383     {
2384         \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2385         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2386         \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2387         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2388     }
2389     \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2390     \cs_set:Npn \__enumext_tmp:n ##1
2391     { . \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.

```

2392     \bool_lazy_all:nT
2393     {
2394         { \bool_if_p:N \g__enumext_starred_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         { \tl_use:N \l__enumext_label_copy_vii_tl }
2400     }

```

```

2401 \bool_lazy_all:nT
2402 {
2403   { \bool_not_p:n { \g__enumext_standar_bool } }
2404   { \bool_if_p:N \l__enumext_standar_bool }
2405   { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } } }
2406 }
2407 {
2408   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2409   {
2410     \tl_use:N \l__enumext_label_copy_vii_tl
2411     \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2412   }
2413 }

```

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.

```

2414 \bool_lazy_all:nT
2415 {
2416   { \bool_if_p:N \g__enumext_standar_bool }
2417   { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } } }
2418 { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } } }
2419 }
2420 {
2421   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2422   {
2423     \tl_use:N \l__enumext_label_copy_i_tl
2424     \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2425   }
2426 }
2427 \cs_set:Npn \__enumext_tmp:n ##1
2428 { \tl_use:c { \l__enumext_label_copy_ \int_to_roman:n {##1} _tl } . }
2429 \bool_lazy_all:nT
2430 {
2431   { \bool_if_p:N \g__enumext_standar_bool }
2432   { \bool_if_p:N \l__enumext_starred_bool }
2433   { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } } }
2434 }
2435 {
2436   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2437   {
2438     \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2439     \tl_use:N \l__enumext_label_copy_vii_tl
2440   }
2441 }

```

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

```

2442 \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2443 {
2444   \l__enumext_store_name_tl \c_colon_str
2445   \int_eval:n { \prop_count:c { \g__enumext_ \l__enumext_store_name_tl _prop } }
2446 }

```

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.

```

2447 \tl_put_right:Ne \l__enumext_write_aux_file_tl
2448 {
2449   \__enumext_newlabel:nn
2450   { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2451   { \l__enumext_newlabel_arg_two_tl }
2452 }
2453 \l__enumext_write_aux_file_tl
2454 }

```

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

```

2455 \cs_new_protected:Npn \__enumext_store_anskey_code:n #1
2456 {

```

```

2457 \int_gincr:N \g__enumext_item_anskey_int
2458 \__enumext_store_addto_prop:n {#1}
2459 \bool_if:NT \l__enumext_store_ref_key_bool
2460 {
2461   \__enumext_store_internal_ref:
2462 }
2463 \__enumext_anskey_show_wrap_left:n { #1 }

```

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

```

2464 \tl_clear:N \l__enumext_store_anskey_arg_tl
2465 \bool_lazy_and:nnT
2466 { \bool_if_p:N \l__enumext_store_columns_break_bool }
2467 { \bool_not_p:n { \l__enumext_starred_bool } }
2468 {
2469   \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2470 }
2471 \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { \item }

```

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

```

2472 \bool_lazy_and:nnT
2473 { \bool_not_p:n { \l__enumext_starred_bool } }
2474 { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2475 {
2476   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2477   {
2478     ( \exp_not:V \l__enumext_store_item_join_int )
2479   }
2480 }

```

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

```

2481 \bool_if:NTF \l__enumext_store_item_star_bool
2482 {
2483   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2484   \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2485   {
2486     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2487     {
2488       [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2489     }
2490   }
2491   \dim_compare:nT
2492   {
2493     \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2494   }
2495   {
2496     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2497     {
2498       [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2499     }
2500   }
2501   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2502 }
2503 {
2504   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2505 }

```

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

```

2506 \bool_lazy_and:nnT
2507 { \bool_if_p:N \l__enumext_store_ref_key_bool }
2508 { \bool_if_p:N \l__enumext_hyperref_bool }
2509 {
2510   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2511   {
2512     \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2513     { \exp_not:V \l__enumext_mark_ref_sym_tl }
2514   }

```

```

2515     }
2516     \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2517 }

```

(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.

```

2518 \cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
2519 {
2520   \par
2521   \bool_if:NTF \l__enumext_starred_bool
2522   {
2523     \__enumext_print_keyans_box:NN \l__enumext_labelwidth_vii_dim \l__enumext_labelsep_vii_dim
2524   }
2525   {
2526     \__enumext_print_keyans_box:cc
2527     { \l__enumext_labelwidth_ \l__enumext_level: _dim }
2528     { \l__enumext_labelsep_ \l__enumext_level: _dim }
2529   }
2530   \__enumext_anskey_wrapper:n { #1 }
2531 }

```

(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.

```

2532 \cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
2533 {
2534   \bool_if:NT \l__enumext_show_answer_bool
2535   {
2536     \__enumext_anskey_show_wrap_arg:n { #1 }
2537   }
2538   \bool_if:NT \l__enumext_show_position_bool
2539   {
2540     \tl_set:Nx \l__enumext_mark_answer_sym_tl
2541     {
2542       \group_begin:
2543       \exp_not:N \normalfont
2544       \exp_not:N \footnotesize [ \int_eval:n
2545       {
2546         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2547       }
2548       ]
2549       \group_end:
2550     }
2551     \__enumext_anskey_show_wrap_arg:n { #1 }
2552   }
2553 }

```

(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

```

2554 \keys_define:nn { enumext / anskey }
2555 {
2556   break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2557   break-col .default:n = true,
2558   break-col .value_forbidden:n = true,
2559   item-join .int_set:N = \l__enumext_store_item_join_int,
2560   item-join .value_required:n = true,
2561   item-star .bool_set:N = \l__enumext_store_item_star_bool,
2562   item-star .default:n = true,

```

```

2563     item-star .value_forbidden:n = true,
2564     item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
2565     item-sym* .value_required:n = true,
2566     item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
2567     item-pos* .value_required:n = true,
2568     unknown .code:n = { \l__enumext_anskey_unknown:n {#1} },
2569 }

```

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`.

```

2570 \cs_new_protected:Npn \l__enumext_anskey_unknown:n #1
2571 {
2572     \exp_args:NV \l__enumext_anskey_unknown:nn \l_keys_key_str {#1}
2573 }
2574 \cs_new_protected:Npn \l__enumext_anskey_unknown:nn #1 #2
2575 {
2576     \tl_if_blank:nTF {#2}
2577     {
2578         \msg_error:nnn { enumext } { anskey-cmd-key-unknown } {#1}
2579     }
2580     {
2581         \msg_error:nnnn { enumext } { anskey-cmd-key-value-unknown } {#1} {#2}
2582     }
2583 }

```

(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`.

```

2584 \NewDocumentCommand \anskey { o +m }
2585 {
2586     \l__enumext_anskey_safe_outer:
2587     \group_begin:
2588         \bool_if:NT \l__enumext_check_answers_bool
2589         {
2590             \tl_if_novalue:nF {#1}
2591             {
2592                 \keys_set:nn { enumext / anskey } {#1}
2593             }
2594             \tl_if_blank:nTF {#2}
2595             {
2596                 \msg_error:nn { enumext } { anskey-empty-arg }
2597             }
2598             {
2599                 \l__enumext_anskey_safe_inner:
2600                 \l__enumext_store_anskey_code:n {#2}
2601             }
2602         }
2603     \group_end:
2604 }

```

(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:`

```

2605 \cs_new_protected:Nn \l__enumext_anskey_safe_outer:
2606 {
2607     \bool_if:NF \l__enumext_store_active_bool
2608     {
2609         \msg_error:nnnn { enumext } { anskey-wrong-place } { anskey } { enumext }
2610     }
2611     \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
2612     {
2613         \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans }
2614     }
2615 }

```



```

2614     }
2615     \int_compare:nNt { \__enumext_keyans_level_h_int } = { 1 }
2616     {
2617         \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans* }
2618     }
2619     \int_compare:nNt { \__enumext_keyans_pic_level_int } = { 1 }
2620     {
2621         \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyanspic }
2622     }
2623 }

```

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.

```

2624 \cs_new_protected:Nn \__enumext_anskey_safe_inner:
2625 {
2626     \int_incr:N \__enumext_anskey_level_int
2627     \int_compare:nNt { \__enumext_anskey_level_int } > { 1 }
2628     {
2629         \msg_error:nn { enumext } { anskey-nested }
2630     }
2631     \bool_if:NF \__enumext_item_number_bool
2632     {
2633         \msg_error:nn { enumext } { anskey-unnumber-item }
2634     }
2635     \mode_if_math:T
2636     {
2637         \msg_error:nne { enumext } { anskey-math-mode } { \c_backslash_str anskey }
2638     }
2639 }

```

(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.

```

2640 \cs_new_protected:Nn \__enumext_undefine_anskey_env:
2641 {
2642     \cs_undefine:c { anskey* }
2643     \cs_undefine:c { endanskey* }
2644     \cs_undefine:c { __scontents_anskey*_env_begin: }
2645     \cs_undefine:c { __scontents_anskey*_env_end: }
2646 }

```

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

```

2647 \__enumext_before_env:nn { enumext }
2648 {
2649     \bool_lazy_and:nnT
2650     { \int_compare_p:nNn { \__enumext_level_int } = { 0 } }
2651     { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
2652     {
2653         \cs_if_free:cF { __scontents_anskey*_env_begin: }
2654         {
2655             \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2656         }
2657     }

```

```

2658   }
2659   \__enumext_before_env:nn { enumext* }
2660   {
2661     \bool_lazy_and:nnT
2662     { \int_compare_p:nNn { \__enumext_level_int } = { 0 } }
2663     { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
2664     {
2665       \cs_if_free:cF { __scontents_anskey*_env_begin: }
2666       {
2667         \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2668       }
2669     }
2670   }

```

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.

```

2671   \__enumext_before_env:nn { anskey* }
2672   {
2673     \int_compare:nNnT { \__enumext_keyans_level_int } = { 1 }
2674     {
2675       \msg_error:nnn { enumext } { anskey-env-wrong } { keyans }
2676     }
2677     \int_compare:nNnT { \__enumext_keyans_level_h_int } = { 1 }
2678     {
2679       \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
2680     }
2681     \int_compare:nNnT { \__enumext_keyans_pic_level_int } = { 1 }
2682     {
2683       \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
2684     }
2685     \bool_if:NF \__enumext_item_number_bool
2686     {
2687       \msg_error:nn { enumext } { anskey-unnumber-item }
2688     }
2689     \mode_if_math:T
2690     {
2691       \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
2692     }
2693   }

```

(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`.

```

2694   \cs_new_protected:Npn \__enumext_anskey_env_make:n #1
2695   {
2696     \bool_if:NT \__enumext_anskey_env_bool
2697     {
2698       \newenvsc{anskey*}[store-env=#1,print-env=false]
2699       \__enumext_anskey_env_exec:
2700     }
2701   }
2702   \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`.

```

2703   \cs_new_protected:Nn \__enumext_anskey_env_define_keys:
2704   {
2705     \keys_define:nn { scontents / scontents }
2706     {
2707       break-col .bool_gset:N = \g__enumext_store_columns_break_bool,
2708       break-col .default:n = true,
2709       break-col .value_forbidden:n = true,
2710       item-join .int_gset:N = \g__enumext_store_item_join_int,
2711       item-join .value_required:n = true,

```

```

2712     item-star .bool_gset:N = \g__enumext_store_item_star_bool,
2713     item-star .default:n   = true,
2714     item-star .value_forbidden:n = true,
2715     item-sym* .tl_gset:N   = \g__enumext_store_item_symbol_tl,
2716     item-sym* .value_required:n = true,
2717     item-pos* .dim_gset:N   = \g__enumext_store_item_symbol_sep_dim,
2718     item-pos* .value_required:n = true,
2719     print-env .undefine:,
2720     store-env .undefine:,
2721     write-out .undefine:,
2722     unknown   .code:n      = { \__enumext_anskey_env_unknown:n {##1} },
2723   }
2724 }

```

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`.

```

2725 \cs_new_protected:Npn \__enumext_anskey_env_unknown:n #1
2726 {
2727   \exp_args:NV \__enumext_anskey_env_unknown:nn \l_keys_key_str {#1}
2728 }
2729 \cs_new_protected:Npn \__enumext_anskey_env_unknown:nn #1#2
2730 {
2731   \tl_if_blank:nTF {#2}
2732   {
2733     \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
2734   }
2735   {
2736     \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
2737   }
2738 }

```

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`.

```

2739 \cs_new_protected:Nn \__enumext_anskey_env_reset_keys:
2740 {
2741   \keys_define:nn { scontents / scontents }
2742   {
2743     break-col .undefine:,
2744     item-join .undefine:,
2745     item-star .undefine:,
2746     item-sym* .undefine:,
2747     item-pos* .undefine:,
2748     write-out .code:n   = {
2749       \bool_set_false:N \l__scontents_storing_bool
2750       \bool_set_true:N  \l__scontents_writing_bool
2751       \tl_set:Nn \l__scontents_fname_out_tl {##1}
2752     },
2753     write-out .value_required:n = true,
2754     print-env .meta:nn         = { scontents } { print-env = ##1 },
2755     print-env .default:n       = true,
2756     store-env .meta:nn         = { scontents } { store-env = ##1 },
2757     unknown   .code:n          = { \__scontents_parse_environment_keys:n {##1} },
2758   }
2759 }

```

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*.

```

2760 \cs_new_protected:Npn \__enumext_rescan_anskey_env:n #1
2761 {
2762   \group_begin:
2763     \int_set:Nn \tex_newlinechar:D { `^^J }
2764     \__scontents_rescan_tokens:x
2765     {
2766       \endgroup % This assumes \catcode`\=0... Things might go off otherwise.
2767       #1
2768     }
2769 }

```

(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)`.

```
2770 \cs_new_protected:Nn \__enumext_anskey_env_exec:
2771 {
2772   \__enumext_before_env:nn { anskey* }
2773   {
2774     \__enumext_anskey_env_define_keys:
2775   }
```

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.

```
2776   \hook_if_empty:nF {env/anskey*/after}
2777   {
2778     \hook_gremove_code:nn {env/anskey*/after} { * }
2779   }
2780   \__enumext_after_env:nn { anskey* }
2781   {
2782     \__enumext_anskey_env_save_keys:
2783     \tl_clear:N \l__enumext_store_anskey_env_tl
2784     \tl_clear:N \l__enumext_store_anskey_opt_tl
2785     \bool_if:NT \l__enumext_check_answers_bool
2786     {
2787       \tl_gset:Ne \l__enumext_store_anskey_env_tl
2788       {
2789         \seq_item:ce { g__scontents_name_ \l__enumext_store_name_tl _seq } { -1 }
2790       }
2791       \regex_match:nVTF
2792       { ^\s* \z | ^\s* \u{c__scontents_hidden_space_str} \z }
2793       \l__enumext_store_anskey_env_tl
2794       {
2795         \msg_error:nn { enumext } { anskey-empty-arg }
2796       }
2797       {
2798         \__enumext_anskey_env_store:
2799       }
2800     }
2801     \__enumext_anskey_env_clean_vars:
2802     \__enumext_anskey_env_reset_keys:
2803   }
2804 }
```

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:`

```
2805 \cs_new_protected:Nn \__enumext_anskey_env_save_keys:
2806 {
2807   \bool_lazy_and:nnT
2808   { \bool_if_p:N \g__enumext_store_columns_break_bool }
2809   { \bool_not_p:n { \l__enumext_starred_bool } }
2810   {
2811     \tl_put_left:Ne \l__enumext_store_anskey_opt_tl { ,break-col, }
2812   }
```

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

```
2813   \bool_lazy_and:nnT
2814   { \bool_not_p:n { \l__enumext_starred_bool } }
2815   { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }
2816   {
2817     \tl_put_left::Ne \l__enumext_store_anskey_opt_tl
2818     {
2819       ,item-join = \exp_not:V \g__enumext_store_item_join_int,
2820     }
2821   }
```

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

```

2822 \bool_if:NT \g__enumext_store_item_star_bool
2823 {
2824   \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2825   {
2826     ,item-star,
2827   }
2828   \tl_if_empty:NF \g__enumext_store_item_symbol_tl
2829   {
2830     \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2831     {
2832       ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,
2833     }
2834   }
2835   \dim_compare:nT
2836   {
2837     \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
2838   }
2839   {
2840     \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2841     {
2842       ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
2843     }
2844   }
2845 }
2846 }

```

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`.

```

2847 \cs_new_protected:Nn \__enumext_anskey_env_store:
2848 {
2849   \group_begin:
2850   \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
2851   {
2852     \exp_args:Ne
2853     \__enumext_store_anskey_code:n
2854     {
2855       \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2856     }
2857   }
2858   {
2859     \keys_set_known:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
2860     \exp_args:Ne
2861     \__enumext_store_anskey_code:n
2862     {
2863       \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2864     }
2865   }
2866   \group_end:
2867 }

```

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

```

2868 \cs_new_protected:Nn \__enumext_anskey_env_clean_vars:
2869 {
2870   \bool_gset_false:N \g__enumext_store_columns_break_bool
2871   \int_gzero:N \g__enumext_store_item_join_int
2872   \bool_gset_false:N \g__enumext_store_item_star_bool
2873   \tl_gclear:N \g__enumext_store_item_symbol_tl
2874   \dim_gzero:N \g__enumext_store_item_symbol_sep_dim
2875 }

```

(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.

```

2876 \cs_new_protected:Nn \__enumext_execute_after_env:
2877 {
2878   \int_compare:nNtT { \__enumext_level_int } = { 0 }
2879   {
2880     \tl_if_empty:NF \g__enumext_store_name_tl
2881     {
2882       \__enumext_stop_save_ans_msg:
2883       \__enumext_item_answer_diff:
2884       \__enumext_log_global_vars:
2885       \__enumext_log_answer_vars:
2886       \bool_if:NTF \g__enumext_check_ans_key_bool
2887       {
2888         \__enumext_check_ans_show:
2889       }
2890       { \__enumext_check_ans_log: }
2891       \__enumext_undefine_anskey_env:
2892     }
2893     \__enumext_reset_global_vars:
2894   }
2895 }

```

(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`.

```

2896 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2897 {
2898   \tl_clear:N \l__enumext_store_current_label_tl
2899   \int_compare:nNtTF { \__enumext_keyans_pic_level_int } = { 1 }
2900   {
2901     \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
2902   }
2903   {
2904     \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
2905   }
2906   \tl_if_novalue:NF { #1 }
2907   {
2908     % Set save-sep
2909     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2910     {
2911       \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
2912     }
2913     \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2914   }
2915   \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
2916 }

```

(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.

```

2917 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2918 {
2919   \bool_if:NT \l__enumext_store_ref_key_bool
2920   {
2921     \cs_set_protected:Npn \__enumext_tmp:n ##1
2922     {
2923       \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2924       \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2925       \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2926       \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2927     }
2928     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2929     \__enumext_keyans_store_ref_aux_i:
2930   }
2931 }

```

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`.

```

2932 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2933 {
2934   \bool_if:NT \g__enumext_starred_bool
2935   {
2936     \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2937   }
2938   \int_compare:nNt { \l__enumext_keyans_pic_level_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_vi_tl }
2942   }
2943   \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
2944   {
2945     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2946     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2947   }
2948   \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
2949   {
2950     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2951     { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2952   }
2953   \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2954   {
2955     \l__enumext_store_name_tl \c_colon_str
2956     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2957   }
2958   \__enumext_keyans_store_ref_aux_ii:
2959 }

```

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.

```

2960 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2961 {
2962   \tl_put_right:Ne \l__enumext_write_aux_file_tl
2963   {
2964     \__enumext_newlabel:nn
2965     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2966     { \l__enumext_newlabel_arg_two_tl }
2967   }
2968   \l__enumext_write_aux_file_tl
2969 }

```

(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.

```

2970 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1

```



```

2971 {
2972   \tl_clear:N \l__enumext_store_current_label_tl
2973   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2974   {
2975     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
2976   }
2977   {
2978     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
2979   }
2980   \tl_if_novalue:nF { #1 }
2981   {
2982     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2983     {
2984       \tl_put_right:Ne \l__enumext_store_current_label_tl
2985       {
2986         \l__enumext_store_keyans_item_opt_sep_tl
2987       }
2988     }
2989     \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2990   }
2991   \__enumext_keyans_addto_seq_link:
2992 }

```

Checks if the `save-ref` key is active along with the `hyperref` package load, if both conditions are met, it will create the `\hyperlink` and then store using the `__enumext_store_addto_seq:V` function. Finally, copy the contents of the variable `\l__enumext_store_current_label_tl` into the global variable `\g__enumext_check_ans_item_tl` to be used by the function `__enumext_check_starred_cmd:n` and increment the value of the integer variable `\g__enumext_item_anskey_int` handled by the `check-ans` key.

```

2993 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2994 {
2995   \bool_lazy_and:nnT
2996   { \bool_if_p:N \l__enumext_store_ref_key_bool }
2997   { \bool_if_p:N \l__enumext_hyperref_bool }
2998   {
2999     \tl_put_right:Ne \l__enumext_store_current_label_tl
3000     {
3001       \hfill \exp_not:N \hyperlink
3002       {
3003         \exp_not:V \l__enumext_newlabel_arg_one_tl
3004       }
3005       { \exp_not:V \l__enumext_mark_ref_sym_tl }
3006     }
3007   }
3008   \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
3009   \bool_if:NT \l__enumext_check_answers_bool
3010   {
3011     \int_gincr:N \g__enumext_item_anskey_int
3012   }
3013 }

```

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

12.32.4 The show-ans and show-pos keys for keyans and keyanspic

The code is very similar to the `\anskey` code, but, if I change the order of the operations the counter off `\label` are incorrect.

```

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

```

Common function to show *starred commands* `\item*` and `\position` of stored content in `\prop list` for `keyans` and `keyanspic`. Need add `1` to `\g__enumext_⟨store name⟩_prop` for `show-pos` key.

```

3014 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
3015 {
3016   \tl_if_novalue:nF { #1 }
3017   {
3018     \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
3019   }
3020   \bool_if:NT \l__enumext_show_answer_bool
3021   {
3022     \__enumext_keyans_show_ans:
3023   }
3024   \bool_if:NT \l__enumext_show_position_bool
3025   {

```

```

3026         \__enumext_keyans_show_pos:
3027     }
3028 }
3029 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
3030 {
3031     \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
3032     {
3033         \bool_lazy_or:nnT
3034         { \bool_if_p:N \l__enumext_show_answer_bool }
3035         { \bool_if_p:N \l__enumext_show_position_bool }
3036         {
3037             \__enumext_keyans_wrapper_opt:n { \l__enumext_store_current_opt_arg_tl } \c_space_tl
3038         }
3039     }
3040 }
3041 \cs_new_protected:Nn \__enumext_keyans_show_ans:
3042 {
3043     \bool_if:NT \l__enumext_starred_bool
3044     {
3045         \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
3046         \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
3047     }
3048     \tl_put_left:Nn \l__enumext_label_v_tl
3049     {
3050         \__enumext_print_keyans_box:NN
3051         \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3052     }
3053 }
3054 \cs_new_protected:Nn \__enumext_keyans_show_pos:
3055 {
3056     \bool_if:NT \l__enumext_starred_bool
3057     {
3058         \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
3059         \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
3060     }
3061     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
3062     {
3063         \tl_set:Ne \l__enumext_mark_answer_sym_tl
3064         {
3065             \group_begin:
3066             \exp_not:N \normalfont
3067             \exp_not:N \footnotesize [ \int_eval:n
3068                 {
3069                     \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3070                 }
3071             ]
3072             \group_end:
3073         }
3074     }
3075     {
3076         \tl_set:Ne \l__enumext_mark_answer_sym_tl
3077         {
3078             \group_begin:
3079             \exp_not:N \normalfont
3080             \exp_not:N \footnotesize [ \int_eval:n
3081                 {
3082                     \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
3083                 }
3084             ]
3085             \group_end:
3086         }
3087     }
3088     \tl_put_left:Nn \l__enumext_label_v_tl
3089     {
3090         \__enumext_print_keyans_box:NN
3091         \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3092     }
3093 }

```

(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`.

```

3094 \cs_new_protected:Npn \__enumext_default_item:n #1
3095 {
3096   \tl_if_novalue:nTF {#1}
3097   {
3098     \bool_if:NT \l__enumext_check_answers_bool
3099     {
3100       \int_gincr:N \g__enumext_item_number_int
3101       \bool_set_true:N \l__enumext_item_number_bool
3102     }
3103     \bool_set_true:c { \l__enumext_wrap_label_ \__enumext_level: _bool }
3104     \__enumext_item_std:w \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
3105   }
3106   {
3107     \bool_set_eq:cc
3108     { \l__enumext_wrap_label_ \__enumext_level: _bool }
3109     { \l__enumext_wrap_label_opt_ \__enumext_level: _bool }
3110     \__enumext_item_std:w [#1] \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
3111   }
3112 }

```

(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`.

```

3113 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
3114 {
3115   \tl_if_novalue:nTF {#1}
3116   {
3117     \tl_gset_eq:Nc
3118     \g__enumext_item_symbol_aux_tl { \l__enumext_item_symbol_ \__enumext_level: _tl }
3119   }
3120   {
3121     \tl_gset:Nn \g__enumext_item_symbol_aux_tl {#1}
3122   }
3123   \tl_if_novalue:nTF {#2}
3124   {
3125     \dim_set_eq:cc
3126     { \l__enumext_item_symbol_sep_ \__enumext_level: _dim }
3127     { \l__enumext_labelsep_ \__enumext_level: _dim }
3128   }
3129   {
3130     \dim_set:cn { \l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
3131   }
3132   \bool_if:NT \l__enumext_check_answers_bool
3133   {
3134     \int_gincr:N \g__enumext_item_number_int
3135     \bool_set_true:N \l__enumext_item_number_bool

```

```

3136     }
3137     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
3138     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3139 }

```

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

```

3140 \cs_new_protected:Nn \__enumext_item_star_exec:
3141 {
3142   \tl_if_empty:cF { l__enumext_item_symbol_ \__enumext_level: _tl }
3143   {
3144     \mode_leave_vertical:
3145     \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3146     \hbox_overlap_left:n { \g__enumext_item_symbol_aux_tl }
3147     \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3148   }
3149 }

```

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

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

```

3150 \cs_new_protected:Nn \__enumext_redefine_item:
3151 {
3152   \RenewDocumentCommand \item { s o o }
3153   {
3154     \bool_if:nTF {##1}
3155     {
3156       \__enumext_starred_item:nn {##2} {##3}
3157     }
3158     { \__enumext_default_item:n {##2} }
3159   }
3160 }

```

(End of definition for `__enumext_redefine_item:`.)

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

```

\__enumext_make_label_std:
\__enumext_make_label_box:
3161 \cs_new_protected:Nn \__enumext_make_label:
3162 {
3163   \IfDocumentMetadataTF
3164   {
3165     \__enumext_make_label_box:
3166   }
3167   { \__enumext_make_label_std: }
3168 }
3169 \cs_new_protected:Nn \__enumext_make_label_std:
3170 {
3171   \RenewDocumentCommand \makeLabel { m }
3172   {
3173     \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
3174     \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3175     \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3176     {
3177       \__enumext_item_star_exec:
3178       \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3179     }
3180     { ##1 }
3181     \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
3182     \tl_gclear:N \g__enumext_item_symbol_aux_tl
3183   }
3184 }
3185 \cs_new_protected:Nn \__enumext_make_label_box:
3186 {
3187   \RenewDocumentCommand \makeLabel { m }
3188   {
3189     \makebox
3190     [ \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim } ]
3191     [ \str_use:c { l__enumext_align_label_pos_ \__enumext_level: _str } ]
3192     {
3193       \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }

```

```

3194         \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3195         {
3196             \__enumext_item_star_exec:
3197             \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3198         }
3199         { ##1 }
3200     \tl_gclear:N \g__enumext_item_symbol_aux_tl
3201 }
3202 }
3203 }

```

(End of definition for `__enumext_make_label:`, `__enumext_make_label_std:`, and `__enumext_make_label_box:`.)

• 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*
3204 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
3205 {
3206     \keys_define:nn { enumext / #1 }
3207     {
3208         item-sym* .tl_set:c = { l__enumext_item_symbol_#2_tl },
3209         item-sym* .value_required:n = true,
3210         item-sym* .initial:n = {$\star$},
3211         item-pos* .dim_set:c = { l__enumext_item_symbol_sep_#2_dim },
3212         item-pos* .value_required:n = true,
3213     }
3214 }
3215 \clist_map_inline:nn
3216 {
3217     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
3218 }
3219 { \__enumext_tmp:nn #1 }

```

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

12.35 Handling unknown keys

At this point in the code I already know that I will not add more `<keys>` and since I have already been quite *paranoid and restrictive* with the definitions of environments and commands, the only thing left to do is do it with the `<keys>` (you have to be consistent in life).

12.35.1 Handling unknown keys for keyans and keyans*

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

```

\__enumext_keyans_unknown_keys:n
\__enumext_keyans_unknown_keys:nn
3220 \cs_set_protected:Npn \__enumext_tmp:n #1
3221 {
3222     \keys_define:nn { enumext / #1 }
3223     {
3224         unknown .code:n = { \__enumext_keyans_unknown_keys:n {##1} }
3225     }
3226 }
3227 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }

```

Internal functions for handling `unknown` key.

```

3228 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:n #1
3229 {
3230     \exp_args:NV \__enumext_keyans_unknown_keys:nn \l_keys_key_str {#1}
3231 }
3232 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:nn #1#2
3233 {
3234     \tl_if_blank:nTF {#2}
3235     {
3236         \msg_error:nnn { enumext } { keyans-unknown-key } {#1}
3237     }
3238     {
3239         \msg_error:nnnn { enumext } { keyans-unknown-key-value } {#1} {#2}
3240     }
3241 }

```

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

12.35.2 Handling unknown keys for enumext*

unknown

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

```
\__enumext_starred_unknown_keys:n
\__enumext_starred_unknown_keys:nn
```

```
3242 \keys_define:nn { enumext / enumext* }
3243 {
3244   unknown .code:n = { \__enumext_starred_unknown_keys:n {#1} }
3245 }
```

Internal functions for handling `unknown` key.

```
3246 \cs_new_protected:Npn \__enumext_starred_unknown_keys:n #1
3247 {
3248   \exp_args:NV \__enumext_starred_unknown_keys:nn \l_keys_key_str {#1}
3249 }
3250 \cs_new_protected:Npn \__enumext_starred_unknown_keys:nn #1#2
3251 {
3252   \tl_if_blank:nTF {#2}
3253   {
3254     \msg_error:nnn { enumext } { starred-unknown-key } {#1}
3255   }
3256   {
3257     \msg_error:nnnn { enumext } { starred-unknown-key-value } {#1} {#2}
3258   }
3259 }
```

(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.

```
\__enumext_standar_unknown_keys:n
\__enumext_standar_unknown_keys:nn
```

```
3260 \cs_set_protected:Npn \__enumext_tmp:n #1
3261 {
3262   \keys_define:nn { enumext / #1 }
3263   {
3264     unknown .code:n = { \__enumext_standar_unknown_keys:n {##1} }
3265   }
3266 }
3267 \clist_map_inline:nn { level-1,level-2,level-3,level-4 } { \__enumext_tmp:n {#1} }
```

Internal functions for handling `unknown` key.

```
3268 \cs_new_protected:Npn \__enumext_standar_unknown_keys:n #1
3269 {
3270   \exp_args:NV \__enumext_standar_unknown_keys:nn \l_keys_key_str {#1}
3271 }
3272 \cs_new_protected:Npn \__enumext_standar_unknown_keys:nn #1#2
3273 {
3274   \tl_if_blank:nTF {#2}
3275   {
3276     \msg_error:nnn { enumext } { standar-unknown-key } {#1}
3277   }
3278   {
3279     \msg_error:nnnn { enumext } { standar-unknown-key-value } {#1} {#2}
3280   }
3281 }
```

(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`.

```
3282 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
3283 {
3284   \tl_if_novalue:nTF { #1 }
3285   {
3286     \bool_set_true:N \__enumext_wrap_label_v_bool
3287     \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl
3288   }
3289   {
3290     \bool_set_eq:NN \__enumext_wrap_label_v_bool \__enumext_wrap_label_opt_v_bool
3291     \__enumext_item_std:w [ #1 ] \tl_use:N \__enumext_fake_item_indent_v_tl
3292   }
3293 }
```

(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>`.

```
3294 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
3295 {
3296   \tl_set_eq:NN \l__enumext_store_current_label_tmp_tl \l__enumext_label_v_tl
3297   \__enumext_keyans_show_left:n { #1 }
3298   \bool_set_true:N \l__enumext_wrap_label_v_bool
3299   \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl \__enumext_keyans_show_item
```

Recover the original value of the current `<label>` and store it first in the `<prop list>` (including the optional argument), run the internal “label and ref” system if the `save-ref` key is active and finally store it in the `<sequence>`.

```
3300   \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_store_current_label_tmp_tl
3301   \__enumext_keyans_addto_prop:n { #1 }
3302   \__enumext_keyans_store_ref:
3303   \__enumext_keyans_addto_seq:n { #1 }
3304   \int_gincr:N \g__enumext_check_starred_cmd_int
3305 }
```

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

`\item*`

`__enumext_keyans_redefine_item:`

`__enumext_keyans_make_label:`

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

```
3306 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
3307 {
3308   \RenewDocumentCommand \item { s o }
3309   {
3310     \bool_if:nTF {##1}
3311     {
3312       \peek_remove_spaces:n
3313       {
3314         \__enumext_keyans_starred_item:n {##2}
3315       }
3316     }
3317     {
3318       \__enumext_keyans_default_item:n {##2}
3319     }
3320   }
3321 }
```

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

`__enumext_keyans_make_label:`

`__enumext_keyans_make_label_std:`

`__enumext_keyans_make_label_box:`

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

```
3322 \cs_new_protected:Nn \__enumext_keyans_make_label:
3323 {
3324   \IfDocumentMetadataTF
3325   {
3326     \__enumext_keyans_make_label_box:
3327   }
3328   { \__enumext_keyans_make_label_std: }
3329 }
3330 \cs_new_protected:Nn \__enumext_keyans_make_label_std:
3331 {
3332   \RenewDocumentCommand \makeLabel { m }
3333   {
3334     \tl_use:N \l__enumext_label_fill_left_v_tl
3335     \tl_use:N \l__enumext_label_font_style_v_tl
3336     \bool_if:NTF \l__enumext_wrap_label_v_bool
3337     {
3338       \__enumext_wrapper_label_v:n { ##1 }
3339     }
3340     { ##1 }
3341     \tl_use:N \l__enumext_label_fill_right_v_tl
```



```

3342     }
3343   }
3344   \cs_new_protected:Nn \__enumext_keyans_make_label_box:
3345   {
3346     \RenewDocumentCommand \makeLabel { m }
3347     {
3348       \makebox[ \l__enumext_labelwidth_v_dim ][ \l__enumext_align_label_pos_v_str ]
3349       {
3350         \tl_use:N \l__enumext_label_font_style_v_tl
3351         \bool_if:NTF \l__enumext_wrap_label_v_bool
3352         {
3353           \__enumext_wrapper_label_v:n { ##1 }
3354         }
3355         { ##1 }
3356       }
3357     }
3358   }

```

(End of definition for `__enumext_keyans_make_label:`, `__enumext_keyans_make_label_std:`, and `__enumext_keyans_make_label_box:`.)

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

12.37 Second argument of the lists

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

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

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

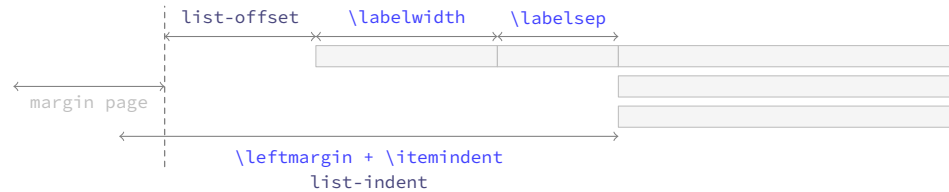


Figure 9: Representation of standard horizontal lengths in `list` environment.

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

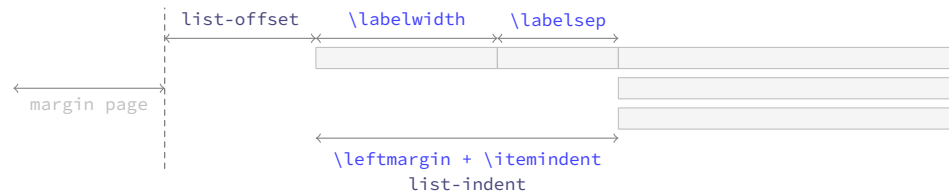


Figure 10: Representation of horizontal lengths concept in list in `enumext`.

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

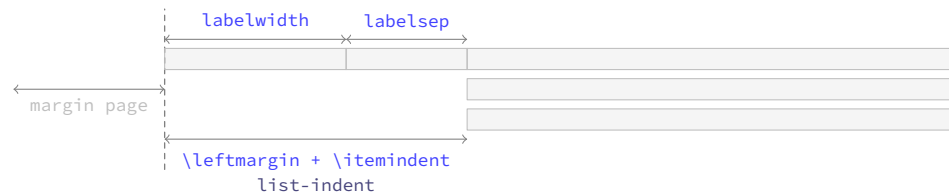


Figure 11: Default horizontal lengths in `enumext`.

```

\__enumext_calc_hspace:NNNNNNN
\__enumext_calc_hspace:ccccccc

```

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

```

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

```

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

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

```

3359 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
3360 {
3361   \dim_compare:nNnT { #1 } < { \c_zero_dim }
3362   {
3363     \msg_warning:nnnV { enumext } { width-non-positive }{ labelwidth }{ #1 }
3364     \dim_set:Nn #1 { \dim_abs:n { #1 } }
3365   }
3366   \dim_compare:nNnT { #2 } < { \c_zero_dim }
3367   {
3368     \msg_warning:nnnV { enumext } { width-negative }{ labelsep }{ #2 }
3369     \dim_set:Nn #2 { \dim_abs:n { #2 } }
3370   }

```

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

```

3371   \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }

```

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

```

3372   \dim_compare:nNnTF { #4 } < { \c_zero_dim }
3373   {
3374     \dim_set:Nn #6 { #1 + #2 - #4 }
3375     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3376   }
3377   {
3378     \dim_compare:nNnT { #4 } = { #1 + #2 }
3379     { \dim_set:Nn #6 { \c_zero_dim } }
3380     \dim_compare:nNnT { #4 } < { #1 + #2 }
3381     { \dim_set:Nn #6 { #1 + #2 - #4 } }
3382     \dim_compare:nNnT { #4 } > { #1 + #2 }
3383     {
3384       \dim_set:Nn #6 { -#1 - #2 + #4 }
3385       \dim_set:Nn #6 { #6*-1 }
3386     }
3387     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3388   }
3389 }
3390 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { cccccc }

```

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

12.37.2 Setting second argument of the lists

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

```

\__enumext_list_arg_two_i:
\__enumext_list_arg_two_ii:
\__enumext_list_arg_two_iii:
\__enumext_list_arg_two_iv:
\__enumext_list_arg_two_v:
3391 \cs_set_protected:Npn \__enumext_tmp:n #1
3392 {
3393   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3394   {
3395     \__enumext_calc_hspace:ccccc
3396     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3397     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3398     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3399     { \__enumext_leftmargin_tmp_#1_bool }
3400     \clist_map_inline:nn
3401     { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3402     { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
3403     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3404     { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
3405     \usecounter { enumX#1 }
3406     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
3407     \str_if_eq:nnTF {#1} { v }
3408     {
3409       \__enumext_keyans_redefine_item:
3410       \__enumext_keyans_make_label:
3411       \__enumext_keyans_ref:
3412       \__enumext_keyans_fake_item:
3413       \bool_if:cT { \__enumext_show_length_#1_bool }
3414       {

```

```

3415         \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3416     }
3417 }
3418 {
3419     \__enumext_redefine_item:
3420     \__enumext_make_label:
3421     \__enumext_standar_ref:
3422     \__enumext_fake_item:
3423     \bool_if:cT { \__enumext_show_length_#1_bool }
3424     {
3425         \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \__enumext_level_i
3426     }
3427 }
3428 }
3429 }
3430 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

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

`__enumext_list_arg_two_vii:` For the horizontal environments `enumext*` and `keyans*` the implementation is similar, but, the value of `\partopsep` is always `\opt`. At this point we will modify the `parsep` key to make it take the value of the `itemsep` key and later, in the environment definition, we will modify `parindent` to make it set the value of `lisparindent` and `parsep` to set the value of `\parskip` locally.

```

3431 \cs_set_protected:Npn \__enumext_tmp:n #1
3432 {
3433     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3434     {
3435         \bool_set_true:c { \__enumext_leftmargin_tmp_#1_bool }
3436         \dim_zero:c { \__enumext_leftmargin_tmp_#1_dim }
3437         \__enumext_calc_hspace:ccccc
3438         { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3439         { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3440         { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3441         { \__enumext_leftmargin_tmp_#1_bool }
3442     \clist_map_inline:nn
3443     { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3444     { \dim_set_eq:cc {###1} { \__enumext_###1_#1_dim } }
3445     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3446     { \skip_set_eq:cc {###1} { \__enumext_###1_#1_skip } }
3447     \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
3448     \skip_zero:N \partopsep
3449     \usecounter { enumX#1 }
3450     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
3451     \__enumext_starred_ref:
3452     \str_if_eq:nnTF {#1} { vii }
3453     {
3454         \__enumext_fake_item_vii:
3455         \bool_if:cT { \__enumext_show_length_vii_bool }
3456         { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3457     }
3458     {
3459         \__enumext_fake_item_viii:
3460         \bool_if:cT { \__enumext_show_length_#1_bool }
3461         { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3462     }
3463 }
3464 }
3465 \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.

```

3466 \NewDocumentEnvironment{enumext}{0}{}
3467 {
3468     \__enumext_safe_exec:
3469     \__enumext_parse_keys:n {#1}
3470     \__enumext_before_list:
3471     \__enumext_start_store_level:
3472     \__enumext_start_list:nn
3473     { \tl_use:c { \__enumext_label_ \__enumext_level: _tl } }

```

```

3474     {
3475         \use:c { __enumext_list_arg_two_ \__enumext_level: : }
3476         \__enumext_before_keys_exec:
3477     }
3478     \__enumext_set_item_width:
3479     \__enumext_after_args_exec:
3480 }
3481 {
3482     \__enumext_after_list:
3483 }

```

(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.

```

3484 \cs_new_protected:Nn \__enumext_set_item_width:
3485 {
3486     \dim_set:Nn \itemwidth
3487     {
3488         \linewidth
3489     }
3490     \dim_compare:nT
3491     {
3492         \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim } != \c_zero_dim
3493     }
3494     {
3495         \dim_sub:Nn \itemwidth
3496         {
3497             \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3498         }
3499     }
3500 }

```

(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 `\l__enumext_level_int` to restrict nesting of the environment, set `\l__enumext_standar_bool` to “true” and finally call the function `__enumext_is_on_first_level:` which sets `\l__enumext_standar_first_bool` to “true” only if the environment is not nested and we are at the “first level”.

```

3501 \cs_new_protected:Nn \__enumext_safe_exec:
3502 {
3503     \__enumext_internal_mini_page:
3504     \__enumext_is_not_nested:
3505     \int_incr:N \l__enumext_level_int
3506     \int_compare:nNnT { \l__enumext_level_int } > { 4 }
3507     { \msg_fatal:nn { enumext } { list-too-deep } }
3508     \bool_set_true:N \l__enumext_standar_bool
3509     \bool_set_false:N \l__enumext_starred_bool
3510     \__enumext_is_on_first_level:
3511 }

```

(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.

```

3512 \cs_new_protected:Npn \__enumext_parse_keys:n #1
3513 {
3514     \tl_if_novalue:nF {#1}
3515     {
3516         \str_clear:N \l__enumext_series_str
3517         \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
3518         {
3519             \keys_set:nn { enumext / level-1 } {#1}
3520             \__enumext_parse_series:n {#1}

```

```

3521         \__enumext_nested_base_line_fix:
3522     }
3523     {
3524         \exp_args:Ne \keys_set:nn
3525         { enumext / level-\int_use:N \__enumext_level_int } {#1}
3526     }
3527     \__enumext_store_active_keys:n {#1}
3528 }
3529 }

```

(End of definition for __enumext_parse_keys:n.)

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*.

```

3530 \cs_new_protected:Nn \__enumext_start_store_level:
3531 {
3532     \bool_lazy_all:nT
3533     {
3534         { \bool_if_p:N \__enumext_store_active_bool }
3535         { \bool_not_p:n { \__enumext_keyans_env_bool } }
3536         { \bool_if_p:N \g__enumext_standar_bool }
3537     }
3538     {
3539         \int_compare:nNnT { \__enumext_level_int } > { 1 }
3540         {
3541             \bool_set_true:c { \__enumext_store_upper_level_ \__enumext_level: _bool }
3542             \__enumext_store_level_open:
3543         }
3544     }

```

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

```

3545     \bool_lazy_all:nT
3546     {
3547         { \bool_if_p:N \__enumext_store_active_bool }
3548         { \bool_not_p:n { \__enumext_keyans_env_bool } }
3549         { \int_compare_p:nNn { \__enumext_level_h_int } = { 1 } }
3550     }
3551     {
3552         \int_compare:nNnT { \__enumext_level_int } > { 0 }
3553         {
3554             \bool_set_true:c { \__enumext_store_upper_level_ \__enumext_level: _bool }
3555             \__enumext_store_level_open:
3556         }
3557     }
3558 }

```

Close the stored structure.

```

3559 \cs_new_protected:Nn \__enumext_stop_store_level:
3560 {
3561     \bool_if:cT { \__enumext_store_upper_level_ \__enumext_level: _bool }
3562     {
3563         \__enumext_store_level_close:
3564     }
3565 }

```

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

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.

```

3566 \cs_new_protected:Nn \__enumext_before_list:
3567 {
3568     \__enumext_vspace_above:
3569     \__enumext_before_args_exec:
3570     \__enumext_check_ans_active:

```

When the mini-env key is active it will set the value of the __enumext_minipage_right_X_dim to be the width of the __enumext_minipage environment on the “right side”, using this value together with the value of the __enumext_minipage_hsep_X_dim set by the mini-sep key, the value of __enumext_minipage_left_X_dim will be set, which will be the width of __enumext_minipage environment on the “left side”, always having a current \linewidth as maximum width between them.

```

3571     \dim_compare:nNnT

```

```

3572 { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3573 {
3574   \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
3575   {
3576     \linewidth
3577     - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
3578     - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
3579   }

```

The boolean variable `__enumext_minipage_active_X_bool` will be activated and the integer variable `\g__enumext_minipage_stat_int` used by the `\miniright` command will be incremented, then the function `__enumext_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.

```

3580   \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
3581   \int_gincr:N \g__enumext_minipage_stat_int
3582   \__enumext_minipage_add_space:
3583   \noindent
3584   \__enumext_mini_page{ \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
3585 }
3586 \__enumext_multicols_start:
3587 }

```

(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.

```

3588 \cs_new_protected:Nn \__enumext_multicols_start:
3589 {
3590   \int_compare:nNnT
3591   { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3592   {
3593     \dim_compare:nNnT
3594     { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3595     {
3596       \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
3597       {
3598         ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
3599         + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
3600         ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
3601         - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3602       }
3603     }
3604     \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
3605     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3606     {
3607       \dim_zero:N \columnseprule
3608     }
3609   }

```

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.

```

3609   \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3610   {
3611     \skip_zero:N \multicolsep
3612     \__enumext_multi_addvspace:
3613   }
3614   \raggedcolumns
3615   \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
3616 }
3617 }

```

(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.

```

3618 \cs_new_protected:Nn \__enumext_multicols_stop:
3619 {
3620   \int_compare:nNnTF
3621   { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }

```

```

3622     {
3623       \__enumext_stop_list:
3624       \__enumext_stop_store_level:
3625       \end{multicols}
3626       \__enumext_unskip_unkern:
3627       \__enumext_unskip_unkern:
3628       \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
3629     }
3630     {
3631       \__enumext_stop_list:
3632       \__enumext_stop_store_level:
3633     }
3634   }

```

(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.

```

3635 \cs_new_protected:Nn \__enumext_after_list:
3636 {
3637   \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
3638   {
3639     \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
3640     {
3641       \msg_warning:nn { enumext } { missing-miniright }
3642       \miniright
3643     }
3644     \int_gzero:N \g__enumext_minipage_stat_int
3645     \__enumext_unskip_unkern: % remove topsep + [partopsep]
3646     \end__enumext_mini_page
3647   }
3648   {
3649     \__enumext_multicols_stop:
3650   }

```

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.

```

3651   \__enumext_after_stop_list:
3652   \__enumext_check_ans_key_hook:
3653   \__enumext_vspace_below:
3654   \bool_set_false:N \l__enumext_standar_bool
3655   \__enumext_resume_save_counter:
3656 }

```

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`.

```

3657 \__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.

```

3658 \NewDocumentEnvironment{keyans}{0}{}
3659 {
3660   \__enumext_keyans_safe_exec:
3661   \__enumext_keyans_parse_keys:n {#1}
3662   \__enumext_before_list_v:
3663   \__enumext_start_list:nn
3664   { \tl_use:N \l__enumext_label_v_tl }
3665   {
3666     \__enumext_list_arg_two_v:
3667     \__enumext_before_keys_exec_v:

```



```

3668     }
3669     \__enumext_keyans_set_item_width:
3670     \__enumext_after_args_exec_v:
3671 }
3672 {
3673     \__enumext_check_starred_cmd:n { item }
3674     %%\__enumext_stop_list:
3675     \__enumext_after_list_v:
3676 }

```

(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.

```

3677 \cs_new_protected:Nn \__enumext_keyans_set_item_width:
3678 {
3679     \dim_set:Nn \itemwidth
3680     {
3681         \linewidth
3682     }
3683     \dim_compare:nT
3684     {
3685         \l__enumext_listoffset_v_dim != \c_zero_dim
3686     }
3687     {
3688         \dim_sub:Nn \itemwidth
3689         {
3690             \l__enumext_listoffset_v_dim
3691         }
3692     }
3693 }

```

(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.

```

3694 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3695 {
3696     \bool_if:NF \l__enumext_store_active_bool
3697     {
3698         \msg_error:nnnn { enumext } { wrong-place } { keyans } { save-ans }
3699     }
3700     \int_incr:N \l__enumext_keyans_level_int
3701     \bool_set_true:N \l__enumext_keyans_env_bool
3702     \__enumext_keyans_name_and_start:
3703     % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3704     \bool_set_false:N \l__enumext_store_active_bool
3705     \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
3706     {
3707         \msg_error:nn { enumext } { keyans-nested }
3708     }
3709     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3710     {
3711         \msg_error:nn { enumext } { keyans-wrong-level }
3712     }
3713 }

```

(End of definition for `__enumext_keyans_safe_exec:.`)

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

```

3714 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3715 {
3716     \keys_set:nn { enumext / keyans } {#1}
3717 }

```

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

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

```

\__enumext_before_list_v:
\__enumext_keyans_multicols_start:
\__enumext_keyans_multicols_stop:
\__enumext_after_list_v:
3718 \cs_new_protected:Nn \__enumext_before_list_v:
3719 {
3720   \__enumext_vspace_above_v:
3721   \__enumext_before_args_exec_v:
3722   \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3723   {
3724     \dim_set:Nn \l__enumext_minipage_left_v_dim
3725     {
3726       \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3727     }
3728     \bool_set_true:N \l__enumext_minipage_active_v_bool
3729     \int_gincr:N \g__enumext_minipage_stat_int
3730     \__enumext_keyans_minipage_add_space:
3731     \__enumext_mini_page{ \l__enumext_minipage_left_v_dim }
3732   }
3733   \__enumext_keyans_multicols_start:
3734 }
3735 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3736 {
3737   \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3738   {
3739     \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3740     {
3741       \dim_set:Nn \l__enumext_columns_sep_v_dim
3742       {
3743         (
3744           \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3745         ) / \l__enumext_columns_v_int
3746         - \l__enumext_listoffset_v_dim
3747       }
3748     }
3749     \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim
3750     \dim_zero:N \columnseprule % no rule here
3751     \bool_if:NF \l__enumext_minipage_active_v_bool
3752     {
3753       \skip_zero:N \multicolsep
3754       \__enumext_keyans_multi_addvspace:
3755     }
3756     \raggedcolumns
3757     \begin{multicols}{\l__enumext_columns_v_int}
3758   }
3759 }
3760 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3761 {
3762   \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
3763   {
3764     \__enumext_stop_list:
3765     \end{multicols}
3766     \__enumext_unskip_unkern:
3767     \__enumext_unskip_unkern:
3768     \par\addvspace{ \l__enumext_multicols_below_v_skip }
3769   }
3770   {
3771     \__enumext_stop_list:
3772   }
3773 }
3774 \cs_new_protected:Nn \__enumext_after_list_v:
3775 {
3776   \bool_if:NTF \l__enumext_minipage_active_v_bool
3777   {
3778     \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3779     {
3780       \msg_warning:nn { enumext } { missing-miniright }
3781       \miniright
3782     }
3783     \int_gzero:N \g__enumext_minipage_stat_int
3784     \__enumext_unskip_unkern: % remove topsep + [partopsep]
3785     \end__enumext_mini_page
3786     \par\addvspace{ \l__enumext_minipage_after_skip }
3787   }

```

```

3788     {
3789         \__enumext_keyans_multicols_stop:
3790     }
3791     \bool_set_false:N \__enumext_keyans_env_bool
3792     \__enumext_after_stop_list_v:
3793     \__enumext_vspace_below_v:
3794 }

```

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

12.40 Tagging PDF support for non-standart list environments

The \TeX release 2022-06-01 brings automatic support for tagPDF in several aspects, including the standart *list environments* and the `list` environment. Unfortunately non-standard *list environments* like `keyanspic` or the horizontal list environments `enumext*` and `keyans*` are not structured in a nice way, i.e. the expected result in the PDF file is the expected one, but the underlying structure is not correct. In simple terms, for tagPDF a list environment is a list environment, no matter what it looks like in the PDF file.

To maintain a correct list structure when `\DocumentMetadata` is active, it is necessary to do some things manually. This implementation is an adaptation of my answer thanks to Ulrike Fischer's comments in [How can I modify my `\item` redefinition to be compatible with tagging-pdf.](#)

12.40.1 Socket for tagging support in `enumext*` and `keyans*`

We will first define the necessary sockets and their behavior for `enumext*` and `keyans*`.

```

start-list-tags
stop-start-tags
stop-list-tags
__enumext_start_list_tag:n
    \__enumext_stop_start_list_tag:
    \__enumext_stop_list_tag:n
3795 \socket_new:nn {tagsupport/enumext/starred}{ 1 }
3796 \socket_new_plug:nnn {tagsupport/enumext/starred} {start-list-tags}
3797 {
3798     \tag_resume:n {#1}
3799     \tag_struct_begin:n {tag=LI}
3800     \tag_struct_begin:n {tag=Lbl}
3801     \tag_mc_begin:n {tag=Lbl}
3802 }
3803 \socket_new_plug:nnn {tagsupport/enumext/starred} {stop-start-tags}
3804 {
3805     \tag_mc_end:
3806     \tag_struct_end:n {tag=Lbl}
3807     \tag_struct_begin:n {tag=LBody}
3808     \tag_struct_begin:n {tag=text-unit}
3809     \tag_struct_begin:n {tag=text}
3810 }
3811 \socket_new_plug:nnn {tagsupport/enumext/starred} {stop-list-tags}
3812 {
3813     \tag_struct_end:n {tag=text}
3814     \tag_struct_end:n {tag=text-unit}
3815     \tag_struct_end:n {tag=LBody}
3816     \tag_struct_end:n {tag=LI}
3817     \tag_suspend:n {#1}
3818 }

```

And now we'll wrap them so that they're only active when `\DocumentMetadata` is present.

```

3819 \cs_new_protected_nopar:Npn \__enumext_start_list_tag:n #1
3820 {
3821     \IfDocumentMetadataTF
3822     {
3823         \socket_assign_plug:nn {tagsupport/enumext/starred} {start-list-tags}
3824         \socket_use:n {tagsupport/enumext/starred} {#1}
3825     } {}
3826 }
3827 \cs_new_protected_nopar:Nn \__enumext_stop_start_list_tag:
3828 {
3829     \IfDocumentMetadataTF
3830     {
3831         \socket_assign_plug:nn {tagsupport/enumext/starred} {stop-start-tags}
3832         \socket_use:nn {tagsupport/enumext/starred} { }
3833     } {}
3834 }
3835 \cs_new_protected_nopar:Npn \__enumext_stop_list_tag:n #1
3836 {
3837     \IfDocumentMetadataTF
3838     {
3839         \socket_assign_plug:nn {tagsupport/enumext/starred} {stop-list-tags}
3840         \socket_use:nn {tagsupport/enumext/starred} {#1}
3841     } {}

```

```
3842 }
```

(End of definition for `start-list-tags` and others.)

12.40.2 Socket for tagging support in `keyanspic`

We will first define the necessary sockets and their behavior for `keyanspic`.

```
start-list-tags
stop-start-tags 3843 \socket_new:nn {tagsupport/enumext/keyanspic}{ 0 }
stop-list-tags 3844 \socket_new_plug:nnn {tagsupport/enumext/keyanspic} {start-list-tags}
__enumext_anspic_start_list_tag: 3845 {
__enumext_anspic_stop_start_list_tag: 3846 \tag_resume:n {keyanspic}
__enumext_anspic_stop_list_tag: 3847 \tag_struct_begin:n {tag=LI}
3848 \tag_struct_begin:n {tag=Lbl}
3849 \tag_mc_begin:n {tag=Lbl}
3850 }
3851 \socket_new_plug:nnn {tagsupport/enumext/keyanspic} {stop-start-tags}
3852 {
3853 \tag_mc_end:
3854 \tag_struct_end:n {tag=Lbl}
3855 \tag_struct_begin:n {tag=LBody}
3856 \tag_struct_begin:n {tag=text-unit}
3857 \tag_struct_begin:n {tag=text}
3858 \tag_mc_begin:n {tag=text}
3859 }
3860 \socket_new_plug:nnn {tagsupport/enumext/keyanspic} {stop-list-tags}
3861 {
3862 \tag_mc_end:
3863 \tag_struct_end:n {tag=text-unit}
3864 \tag_struct_end:n {tag=text}
3865 \tag_struct_end:n {tag=LBody}
3866 \tag_struct_end:n {tag=LI}
3867 \tag_suspend:n {keyanspic}
3868 }
```

And now we'll wrap them so that they're only active when `\DocumentMetadata` is present.

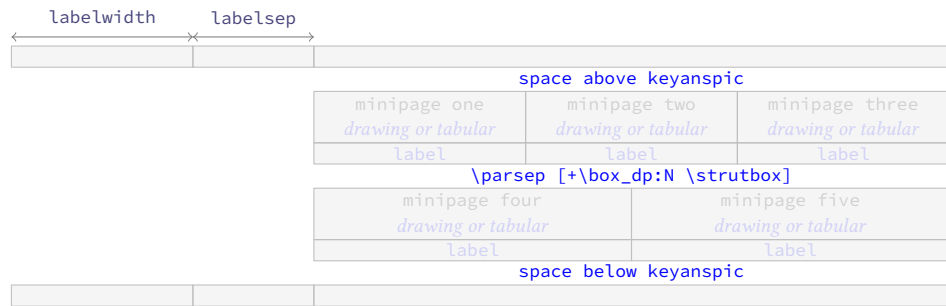
```
3869 \cs_new_protected_nopar:Nn __enumext_anspic_start_list_tag:
3870 {
3871 \IfDocumentMetadataTF
3872 {
3873 \socket_assign_plug:nn {tagsupport/enumext/keyanspic} {start-list-tags}
3874 \socket_use:n {tagsupport/enumext/keyanspic}
3875 } {}
3876 }
3877 \cs_new_protected_nopar:Nn __enumext_anspic_stop_start_list_tag:
3878 {
3879 \IfDocumentMetadataTF
3880 {
3881 \socket_assign_plug:nn {tagsupport/enumext/keyanspic} {stop-start-tags}
3882 \socket_use:n {tagsupport/enumext/keyanspic}
3883 } {}
3884 }
3885 \cs_new_protected_nopar:Nn __enumext_anspic_stop_list_tag:
3886 {
3887 \IfDocumentMetadataTF
3888 {
3889 \socket_assign_plug:nn {tagsupport/enumext/keyanspic} {stop-list-tags}
3890 \socket_use:n {tagsupport/enumext/keyanspic}
3891 } {}
3892 }
```

(End of definition for `start-list-tags` and others.)

12.41 The environment `keyanspic` and `\anspic`

The `keyanspic` environment is a list-based environment that uses the same configuration for “*spacing*” and *label* as the `keyans` environment, but it does not use `\item`. The *contents* are passed to the environment by means of the `\anspic` command and are placed inside `minipage` environments, with the *label* underneath, adjusting widths according to the options passed to the environment.

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

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

12.41.1 The environment `keyanspic`

The environment we wish to build will be based on the `list` environment and will take two optional arguments, the *starred argument* `*` will set the position of the `<label>` to the top if present and bottom otherwise, the second optional argument will process the number of *drawing* or *tabular* that will be on top, bottom or inline when present or not.

`_enumext_keyans_pic_safe_exec:n`

The function `_enumext_keyans_pic_safe_exec:n` check the starred argument and nested level position inside the `enumext` environment. We will set the state of the variable `_enumext_keyans_pic_star_bool` along with the value of the variable `_enumext_keyans_pic_label_pos_str` according to the presence of the starred argument.

```

3893 \\cs_new_protected:Npn \\_enumext_keyans_pic_safe_exec:n #1
3894 {
3895   \\int_incr:N \\_enumext_keyans_pic_level_int
3896   \\int_compare:nNnT { \\_enumext_keyans_pic_level_int } > { 1 }
3897   {
3898     \\msg_error:nn { enumext } { keyanspic-nested }
3899   }
3900   \\_enumext_keyans_name_and_start:
3901   \\bool_if:nTF { #1 }
3902   {
3903     \\bool_set_true:N \\_enumext_keyans_pic_star_bool
3904     \\str_set:Nn \\_enumext_keyans_pic_label_pos_str { t }
3905   }
3906   {
3907     \\str_set:Nn \\_enumext_keyans_pic_label_pos_str { b }
3908   }
3909 }

```

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

`_enumext_keyans_pic_skip_abs:N`

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

```

3910 \\cs_new_protected:Npn \\_enumext_keyans_pic_skip_abs:N #1
3911 {
3912   \\dim_compare:nNnT { #1 } < { \\c_zero_dim }
3913   {
3914     \\skip_set:Nn #1 { -#1 }
3915   }
3916 }

```

(End of definition for `_enumext_keyans_pic_skip_abs:N`.)

`_enumext_keyans_pic_arg_two:`

The `_enumext_keyans_pic_arg_two:` function will be used in the second argument of the `list` environment that defines the `keyanspic` environment, with this we will take the configuration of the spaces and the keys `label` and `wrap-label` from the `keyans` environment.

The first thing we need to do is set the boolean variable `_enumext_leftmargin_tmp_v_bool` handled by the `list-indent` key to false, then copy the definition of the second list argument from the `keyans` environment and make sure that `\\parsep` does not have a negative value.

```

3917 \\cs_new_protected:Npn \\_enumext_keyans_pic_arg_two:
3918 {
3919   \\bool_set_false:N \\_enumext_leftmargin_tmp_v_bool
3920   \\_enumext_list_arg_two_v:
3921   \\_enumext_keyans_pic_skip_abs:N \\parsep

```

Now we increment the `enumXv` counter of the `keyans` environment and save the total height of our `\label` in `\l__enumext_anspic_label_htdp_dim` and we will adjust the values of `\parsep` only if the starred argument is NOT present (default version).

```

3922 \bool_if:NF \l__enumext_keyans_pic_star_bool
3923 {
3924   \stepcounter { enumXv }
3925   \hbox_set:Nn \l__enumext_anspic_label_box { \l__enumext_label_v_tl }
3926   \dim_set:Nn \l__enumext_anspic_label_htdp_dim
3927   {
3928     \box_ht_plus_dp:N \l__enumext_anspic_label_box
3929   }
3930   \skip_add:Nn \parsep
3931   {
3932     \l__enumext_anspic_label_htdp_dim + \box_dp:N \strutbox
3933   }
3934   \skip_gset_eq:NN \g__enumext_keyans_pic_parsep_skip \parsep
3935 }

```

Finally we adjust the value of `\leftmargin` and `\topsep` then set `\labelwidth`, `\labelsep`, `\partopsep` and `\itemsep` to zero so that the horizontal and vertical space is not affected.

```

3936 \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
3937 \skip_add:Nn \topsep { 0.5\box_dp:N \strutbox }
3938 \dim_zero:N \labelwidth
3939 \dim_zero:N \listparindent
3940 \dim_zero:N \labelsep
3941 \skip_zero:N \partopsep
3942 \skip_zero:N \itemsep
3943 }

```

(End of definition for `__enumext_keyans_pic_arg_two:`)

keyanspic Now we define the environment `keyanspic` based on `list`. The starred argument will determine whether the `\labels` will be displayed “above” or “below” the drawing or tabular. The second optional argument [`\number above, number below`] will determine the number of `minipage` environments that will be above and below separated by `\parsep` within it.

```

3944 \NewDocumentEnvironment{keyanspic}{ s o }
3945 {
3946   \__enumext_keyans_pic_safe_exec:n { #1 }
3947   \begin{list} { } { \__enumext_keyans_pic_arg_two: }
3948   \IfDocumentMetadataTF
3949   {
3950     \tag_suspend:n {list}
3951   }{}
3952   \item[] \scan_stop:
3953   % paranoia
3954   \RenewDocumentCommand \item {}
3955   {
3956     \msg_error:nn { enumext } { keyanspic-item-cmd }
3957   }
3958   \IfDocumentMetadataTF
3959   {
3960     \tag_resume:n {keyanspic}
3961     \tag_tool:n {para/tagging=false}
3962     \tag_suspend:n {keyanspic}
3963   } { }
3964 }

```

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

```

3965 {
3966   \IfDocumentMetadataTF
3967   {
3968     \tag_resume:n {keyanspic}
3969     \tag_struct_begin:n {tag=L,attribute=enumerate}
3970   } { }
3971   \tl_if_novalue:nTF { #2 }
3972   {
3973     \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
3974   }

```

```

3975     { \__enumext_keyans_pic_do:n { #2 } }
3976 \IfDocumentMetadataTF
3977 {
3978     \tag_suspend:n {keyanspic}
3979 } { }
3980 \end{list}
3981 \IfDocumentMetadataTF { \tag_struct_end: \tag_struct_end: } { }
3982 \__enumext_check_starred_cmd:n { ansPIC }
3983 \setcounter { enumXvi } { 0 }
3984 \bool_if:NTF \__enumext_keyans_pic_star_bool
3985 {
3986     \par\addvspace{ 0.5\box_dp:N \strutbox }
3987 }
3988 {
3989     \par\addvspace{ \g__enumext_keyans_pic_parsep_skip }
3990 }
3991 %\bool_set_false:N \__enumext_store_active_bool
3992 }

```

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

`__enumext_keyans_pic_do:n` The optional argument is split by comma and is handled directly by the function `__enumext_keyans_pic_do:n` and passed to the function `__enumext_keyans_pic_row:n`.

```

3993 \cs_new_protected:Nn \__enumext_keyans_pic_do:n
3994 {
3995     \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
3996 }
3997 \cs_generate_variant:Nn \__enumext_keyans_pic_do:n { e }

```

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

`__enumext_keyans_pic_row:n` The function `__enumext_keyans_pic_row:n` will set the widths for the `minipage` environments and place the content *stored* by `\ansPIC*` in the `__enumext_keyans_pic_body_seq` sequence inside them.

```

4000 \cs_new_protected:Nn \__enumext_keyans_pic_row:n
4001 {
4002     \dim_set:Nn \__enumext_keyans_pic_width_dim { \linewidth / #1 }
4003     \int_set:Nn \__enumext_keyans_pic_above_int { \__enumext_keyans_pic_below_int }
4004     \int_set:Nn \__enumext_keyans_pic_below_int { \__enumext_keyans_pic_above_int + #1 }
4005     \int_step_inline:nnn
4006     { \__enumext_keyans_pic_above_int + 1 }
4007     { \__enumext_keyans_pic_below_int }
4008     {
4009         \IfDocumentMetadataTF
4010         {
4011             \tag_suspend:n {minipage}
4012         } { }
4013         \begin{minipage}[ \__enumext_keyans_pic_label_pos_str ]{ \__enumext_keyans_pic_width_dim }
4014             \centering
4015             \seq_item:Nn \__enumext_keyans_pic_body_seq { ##1 }
4016         \end{minipage}
4017         \IfDocumentMetadataTF
4018         {
4019             \tag_resume:n {minipage}
4020         } { }
4021     }
4022 \par
4023 }

```

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

12.41.2 The command `\ansPIC`

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

```

4022 \NewDocumentCommand \ansPIC { s o +m }
4023 {

```

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

```

4024     \bool_if:NF \__enumext_store_active_bool
4025     {

```



```

4026         \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
4027     }
4028     \int_compare:nNt { \__enumext_level_int } > { 1 }
4029     {
4030         \msg_error:nn { enumext } { keyanspic-wrong-level }
4031     }
4032     \int_compare:nNt { \__enumext_keyans_level_int } = { 1 }
4033     {
4034         \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
4035     }

```

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.

```

4036     \seq_put_right:Nn \__enumext_keyans_pic_body_seq
4037     {
4038         \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
4039     }
4040 }

```

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

```

__enumext_anspic_box_set_dim:n
__enumext_keyans_anspic_label:nnn
__enumext_keyans_anspic_code:nnn

```

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

```

4041 \cs_new_protected:Npn \__enumext_anspic_box_set_dim:n #1
4042 {
4043     \bool_if:NF \__enumext_keyans_pic_star_bool
4044     {
4045         \IfDocumentMetadataTF
4046         {
4047             \tag_suspend:n {keyanspic}
4048         } { }
4049         \vbox_set:Nn \__enumext_anspic_body_box { #1 }
4050         \dim_set:Nn \__enumext_anspic_body_htdp_dim
4051         {
4052             \box_ht_plus_dp:N \__enumext_anspic_body_box
4053         }
4054         \IfDocumentMetadataTF
4055         {
4056             \tag_resume:n {keyanspic}
4057         } { }
4058     }
4059 }
4060 % process label
4061 \cs_new_protected:Npn \__enumext_anspic_label:nn #1 #2
4062 {
4063     \makebox[ \__enumext_keyans_pic_width_dim ][ c ]
4064     {
4065         \bool_if:Nt { #1 }
4066         {
4067             \__enumext_keyans_addto_prop:n { #2 }
4068             \__enumext_keyans_store_ref:
4069             \__enumext_keyans_addto_seq:n { #2 }
4070             \int_gincr:N \__enumext_check_starred_cmd_int
4071             \bool_lazy_or:nnT
4072             { \bool_if_p:N \__enumext_show_answer_bool }
4073             { \bool_if_p:N \__enumext_show_position_bool }
4074             {
4075                 \tl_set_eq:NN \__enumext_label_v_tl \__enumext_label_vi_tl
4076                 \__enumext_keyans_show_left:n { #2 }
4077                 \tl_set_eq:NN \__enumext_label_vi_tl \__enumext_label_v_tl
4078             }
4079         }
4080         \tl_use:N \__enumext_label_font_style_v_tl
4081         \__enumext_wrapper_label_v:n { \__enumext_label_vi_tl }
4082         \__enumext_keyans_show_item_opt:
4083     }
4084 }
4085 \cs_new_protected:Npn \__enumext_keyans_anspic_label:nnn #1 #2 #3
4086 {
4087     \stepcounter { enumXvi }
4088     \__enumext_anspic_box_set_dim:n { #3 }

```

```

4089 \bool_if:NTF \l__enumext_keyans_pic_star_bool
4090 {
4091   \l__enumext_anspic_label:nn { #1 } { #2 }
4092 }
4093 {
4094   \raisebox
4095   {
4096     -\dim_eval:n
4097     {
4098       \l__enumext_anspic_label_htdp_dim
4099       + \l__enumext_anspic_body_htdp_dim
4100       + \box_dp:N \strutbox
4101     }
4102   }
4103   [ opt ] [ opt ]
4104   {
4105     \l__enumext_anspic_label:nn { #1 } { #2 }
4106   }
4107 }
4108 }
4109 \cs_new_protected:Nn \l__enumext_keyans_anspic_code:nnn
4110 {
4111   \l__enumext_anspic_start_list_tag:
4112   \l__enumext_keyans_anspic_label:nnn { #1 } { #2 } { #3 }
4113   \l__enumext_anspic_stop_start_list_tag:
4114   \\ #3
4115   \l__enumext_anspic_stop_list_tag:
4116 }

```

(End of definition for `\l__enumext_anspic_box_set_dim:n`, `\l__enumext_keyans_anspic_label:nnn`, and `\l__enumext_keyans_anspic_code:nnn`.)

12.42 The horizontal environments

Generating *horizontal list environments* is NOT as simple as standard \LaTeX list environments. The fundamental part of the code is adapted from the `shortlst` package to a more modern version using `expl3`. It is not possible to redefine `\item` and `\makelabel` as in the vertical *non starred* versions.

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 ($\langle number \rangle$).

```

\l__enumext_starred_columns_set_vii:
\l__enumext_starred_columns_set_viii:

```

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

```

4117 \cs_new_protected:Nn \l__enumext_starred_columns_set_vii:
4118 {
4119   \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
4120   {
4121     \dim_set:Nn \l__enumext_columns_sep_vii_dim
4122     {
4123       ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
4124       / \l__enumext_columns_vii_int
4125     }
4126   }
4127   \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - 1 }
4128   \dim_set:Nn \l__enumext_item_width_vii_dim
4129   {
4130     ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
4131     / \l__enumext_columns_vii_int
4132     - \l__enumext_labelwidth_vii_dim
4133     - \l__enumext_labelsep_vii_dim
4134   }

```

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

```

4135 \dim_compare:nNnT { \l__enumext_rightmargin_vii_dim } > { \c_zero_dim }
4136 {
4137   \dim_sub:Nn \l__enumext_item_width_vii_dim
4138   {
4139     ( \l__enumext_rightmargin_vii_dim * \l__enumext_tmpa_vii_int )
4140     / \l__enumext_columns_vii_int

```

```

4141     }
4142     \dim_add:Nn \l__enumext_columns_sep_vii_dim
4143     {
4144         \l__enumext_rightmargin_vii_dim
4145     }
4146 }
4147 }

```

Same implementation for the `keyans*` environment.

```

4148 \cs_new_protected:Nn \l__enumext_starred_columns_set_viii:
4149 {
4150     \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
4151     {
4152         \dim_set:Nn \l__enumext_columns_sep_viii_dim
4153         {
4154             ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
4155             / \l__enumext_columns_viii_int
4156         }
4157     }
4158     \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - 1 }
4159     \dim_set:Nn \l__enumext_item_width_viii_dim
4160     {
4161         ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
4162         / \l__enumext_columns_viii_int
4163         - \l__enumext_labelwidth_viii_dim
4164         - \l__enumext_labelsep_viii_dim
4165     }
4166     \dim_compare:nNnT { \l__enumext_rightmargin_viii_dim } > { \c_zero_dim }
4167     {
4168         \dim_sub:Nn \l__enumext_item_width_viii_dim
4169         {
4170             ( \l__enumext_rightmargin_viii_dim * \l__enumext_tmpa_vii_int )
4171             / \l__enumext_columns_viii_int
4172         }
4173         \dim_add:Nn \l__enumext_columns_sep_viii_dim
4174         {
4175             \l__enumext_rightmargin_viii_dim
4176         }
4177     }
4178 }

```

(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` 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.

```

4179 \cs_new_protected:Npn \l__enumext_starred_joined_item_vii:n #1
4180 {
4181     \int_set:Nn \l__enumext_joined_item_vii_int {#1}
4182     \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
4183     {
4184         \msg_warning:nnee { enumext } { item-joined }
4185         { \int_use:N \l__enumext_joined_item_vii_int }
4186         { \int_use:N \l__enumext_columns_vii_int }
4187         \int_set:Nn \l__enumext_joined_item_vii_int
4188         {
4189             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4190         }
4191     }
4192     \int_compare:nNnT
4193     { \l__enumext_joined_item_vii_int }
4194     >
4195     { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4196     {
4197         \msg_warning:nnee { enumext } { item-joined-columns }
4198         { \int_use:N \l__enumext_joined_item_vii_int }
4199         {
4200             \int_eval:n
4201             { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4202         }

```

```

4203         \int_set:Nn \l__enumext_joined_item_vii_int
4204         {
4205             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4206         }
4207     }
4208     \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { 1 }
4209     {
4210         \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
4211         \int_decr:N \l__enumext_joined_item_aux_vii_int
4212         \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
4213         \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
4214         \dim_set:Nn \l__enumext_joined_width_vii_dim
4215         {
4216             \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
4217             + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
4218                 + \l__enumext_columns_sep_vii_dim
4219             ) * \l__enumext_joined_item_aux_vii_int
4220         }
4221         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
4222     }
4223     {
4224         \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
4225         \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
4226     }
4227 }

```

Same implementation for the `keyans*` environment.

```

4228 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
4229 {
4230     \int_set:Nn \l__enumext_joined_item_viii_int {#1}
4231     \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
4232     {
4233         \msg_warning:nnee { enumext } { item-joined }
4234         { \int_use:N \l__enumext_joined_item_viii_int }
4235         { \int_use:N \l__enumext_columns_viii_int }
4236         \int_set:Nn \l__enumext_joined_item_viii_int
4237         {
4238             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4239         }
4240     }
4241     \int_compare:nNnT
4242     { \l__enumext_joined_item_viii_int }
4243     >
4244     { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4245     {
4246         \msg_warning:nnee { enumext } { item-joined-columns }
4247         { \int_use:N \l__enumext_joined_item_viii_int }
4248         {
4249             \int_eval:n
4250             { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4251         }
4252         \int_set:Nn \l__enumext_joined_item_viii_int
4253         {
4254             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4255         }
4256     }
4257     \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { 1 }
4258     {
4259         \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
4260         \int_decr:N \l__enumext_joined_item_aux_viii_int
4261         \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
4262         \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
4263         \dim_set:Nn \l__enumext_joined_width_viii_dim
4264         {
4265             \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
4266             + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
4267                 + \l__enumext_columns_sep_viii_dim
4268             ) * \l__enumext_joined_item_aux_viii_int
4269         }
4270         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
4271     }
4272     {

```

```

4273         \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
4274         \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
4275     }
4276 }

```

(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

To have compatibility with the `tagPDF` we close the environment according to the presence or not of the `mini-right` key.

```

\__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`.

```

4277 \cs_new_protected:Nn \__enumext_start_mini_vii:
4278 {
4279     \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
4280     {
4281         \dim_set:Nn \l__enumext_minipage_left_vii_dim
4282         {
4283             \linewidth
4284             - \l__enumext_minipage_right_vii_dim
4285             - \l__enumext_minipage_hsep_vii_dim
4286         }
4287         \bool_set_true:N \l__enumext_minipage_active_vii_bool
4288         \dim_gset_eq:NN
4289         \g__enumext_minipage_right_vii_dim
4290         \l__enumext_minipage_right_vii_dim
4291         \__enumext_mini_addvspace_vii:
4292         \nointerlineskip\noindent
4293         \__enumext_mini_page{ \l__enumext_minipage_left_vii_dim }
4294     }
4295 }

```

The function `__enumext_stop_mini_vii:` closes the `__enumext_mini_page` environment on the “left side”, applies `\hfill` and set the variable `\g__enumext_minipage_active_vii_bool` to “true” which will be used in the function `__enumext_after_env:n` to execute the `minipage` on the “right side”.

```

4296 \cs_new_protected:Nn \__enumext_stop_mini_vii:
4297 {
4298     \bool_if:NTF \l__enumext_minipage_active_vii_bool
4299     {
4300         \__enumext_stop_list:
4301         \__enumext_stop_store_level_vii:
4302         \IfDocumentMetadataTF { \tag_resume:n {enumext*} } { }
4303         \end__enumext_mini_page
4304         \hfill
4305         \bool_gset_true:N \g__enumext_minipage_active_vii_bool
4306     }
4307     {
4308         \__enumext_stop_list:
4309         \__enumext_stop_store_level_vii:
4310     }
4311 }

```

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 `minipage` 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.

```

4312 \__enumext_after_env:n {enumext*}
4313 {
4314     \bool_if:NT \g__enumext_minipage_active_vii_bool
4315     {
4316         \__enumext_minipage:w [ t ] { \g__enumext_minipage_right_vii_dim }
4317         \legacy_if_gset_false:n { @minipage }
4318         \skip_vertical:N \c_zero_skip
4319         \par\addvspace { \g__enumext_minipage_right_skip }
4320         \bool_if:NF \g__enumext_minipage_center_vii_bool
4321         {
4322             \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
4323             {

```

```

4324         \centering
4325     }
4326 }
4327 \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
4328 {
4329     \tl_use:N \g__enumext_miniright_code_vii_tl
4330 }
4331 \box_use_drop:N \l__enumext_miniright_code_vii_box
4332 \skip_vertical:N \c_zero_skip
4333 \__enumext_endminipage:
4334 \par\addvspace{ \g__enumext_minipage_after_skip }
4335 }
4336 \bool_gset_false:N \g__enumext_minipage_active_vii_bool
4337 \bool_gset_true:N \g__enumext_minipage_center_vii_bool
4338 \tl_gclear:N \g__enumext_miniright_code_vii_tl
4339 \dim_gzero:N \g__enumext_minipage_right_vii_dim
4340 \bool_gset_false:N \g__enumext_starred_bool
4341 }

```

(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_stop_mini_viii: enumext* environment.

```

4342 \cs_new_protected:Nn \__enumext_start_mini_viii:
4343 {
4344     \dim_compare:nNt { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
4345     {
4346         \dim_set:Nn \l__enumext_minipage_left_viii_dim
4347         {
4348             \linewidth
4349             - \l__enumext_minipage_right_viii_dim
4350             - \l__enumext_minipage_hsep_viii_dim
4351         }
4352         \bool_set_true:N \l__enumext_minipage_active_viii_bool
4353         \dim_gset_eq:NN
4354             \g__enumext_minipage_right_viii_dim
4355             \l__enumext_minipage_right_viii_dim
4356         \__enumext_mini_addvspace_viii:
4357         \nointerlineskip\noindent
4358         \__enumext_mini_page{ \l__enumext_minipage_left_viii_dim }
4359     }
4360 }
4361 \cs_new_protected:Nn \__enumext_stop_mini_viii:
4362 {
4363     \bool_if:NTF \l__enumext_minipage_active_viii_bool
4364     {
4365         \__enumext_stop_list:
4366         \IfDocumentMetadataTF { \tag_resume:n {keyans*} } { }
4367         \end__enumext_mini_page
4368         \hfill
4369         \bool_gset_true:N \g__enumext_minipage_active_viii_bool
4370     }
4371     {
4372         \__enumext_stop_list:
4373     }
4374 }
4375 \__enumext_after_env:nn {keyans*}
4376 {
4377     \bool_if:NT \g__enumext_minipage_active_viii_bool
4378     {
4379         \__enumext_mini_page{ \g__enumext_minipage_right_viii_dim }
4380         \par\addvspace { \g__enumext_minipage_right_skip }
4381         \bool_if:NF \g__enumext_minipage_center_viii_bool
4382         {
4383             \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
4384             {
4385                 \centering
4386             }
4387         }
4388         \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
4389         {

```

```

4390         \tl_use:N \g__enumext_miniright_code_viii_tl
4391     }
4392     \box_use_drop:N \l__enumext_miniright_code_viii_box
4393     \end__enumext_mini_page
4394     \par\addvspace{ \g__enumext_minipage_after_skip }
4395 }
4396 \bool_gset_false:N \g__enumext_minipage_active_viii_bool
4397 \bool_gset_true:N \g__enumext_minipage_center_viii_bool
4398 \tl_gclear:N \g__enumext_miniright_code_viii_tl
4399 \dim_gzero:N \g__enumext_minipage_right_viii_dim
4400 }

```

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

12.42.4 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](#).

```

4401 \cs_new_protected:Nn \__enumext_footnotetext:nn
4402 {
4403     \footnotetext[#1]{#2}
4404 }
4405 \cs_new_protected:Nn \__enumext_renew_footnote:
4406 {
4407     \seq_gclear:N \g__enumext_footnote_arg_seq
4408     \seq_gclear:N \g__enumext_footnote_int_seq
4409     \RenewDocumentCommand \footnote { o +m }
4410     {
4411         \tl_if_novalue:nTF {##1}
4412         {
4413             \stepcounter{footnote}
4414             \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
4415         }
4416         {
4417             \int_gset:Nn \g__enumext_footnote_int { ##1 }
4418         }
4419         \footnotemark [ \g__enumext_footnote_int ]
4420         \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
4421         \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
4422     }
4423 }
4424 \cs_new_protected:Nn \__enumext_print_footnote:
4425 {
4426     \seq_if_empty:NF \g__enumext_footnote_int_seq
4427     {
4428         \seq_map_pairwise_function:NNN
4429         \g__enumext_footnote_int_seq
4430         \g__enumext_footnote_arg_seq
4431         \__enumext_footnotetext:nn
4432     }
4433 }

```

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

12.43 The environment enumext*

`enumext*`

First we will generate the environment and we will give a temporary definition to __enumext_stop_item_tmp_vii: equal to __enumext_first_item_tmp_vii: and next to \item equal to __enumext_start_item_tmp_vii: which we will redefine later. Unlike the implementation used by the `shortlst` package, we will not set the values of \rightskip and \@rightskip equal to \@flushglue whose value is 0.0pt plus 1.0 fil, in the tests I have performed this fails in some circumstances and different results are obtained when using pdfTeX and LuaTeX.

```

4434 \NewDocumentEnvironment{enumext*}{ o }
4435 {
4436     \__enumext_safe_exec_vii:
4437     \__enumext_parse_keys_vii:n {#1}
4438     \__enumext_before_list_vii:
4439     \__enumext_start_store_level_vii:
4440     \__enumext_start_list:nn { }
4441     {
4442         \__enumext_list_arg_two_vii:

```



```

4443     \__enumext_before_keys_exec_vii:
4444   }
4445   % Stop tagging
4446   \IfDocumentMetadataTF { \tag_suspend:n {enumext*} } { }
4447   \__enumext_starred_columns_set_vii:
4448   \item[] \scan_stop:
4449   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_first_item_tmp_vii:
4450   \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
4451   \ignorespaces
4452 }
4453 {
4454   \IfDocumentMetadataTF { \tag_struct_end: } { }
4455   \__enumext_stop_item_tmp_vii:
4456   \__enumext_remove_extra_parsep_vii:
4457   \__enumext_after_list_vii:
4458 }

```

(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.

```

4459 \cs_new_protected:Nn \__enumext_safe_exec_vii:
4460 {
4461   \__enumext_internal_mini_page:
4462   \__enumext_is_not_nested:
4463   \int_incr:N \l__enumext_level_h_int
4464   \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
4465   {
4466     \msg_error:nn { enumext } { nested }
4467   }
4468   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
4469   {
4470     \msg_error:nnn { enumext } { nested-horizontal } { keyans*}
4471   }
4472   \bool_set_true:N \l__enumext_starred_bool
4473   \bool_set_false:N \l__enumext_standar_bool
4474   \__enumext_is_on_first_level:
4475 }

```

(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`.

```

4476 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
4477 {
4478   \tl_if_novalue:nF {#1}
4479   {
4480     \str_clear:N \l__enumext_series_str
4481     \keys_set:nn { enumext / enumext* } {#1}
4482     \__enumext_parse_series:n {#1}
4483     \__enumext_store_active_keys_vii:n {#1}
4484     \__enumext_nested_base_line_fix:
4485   }
4486 }

```

(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*`.

```

4487 \cs_new_protected:Nn \__enumext_before_list_vii:
4488 {

```

```

4489     \__enumext_vspace_above_vii:
4490     \__enumext_check_ans_active:
4491     \__enumext_before_args_exec_vii:
4492     \__enumext_start_mini_vii:
4493 }

```

(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.

```

4494 \cs_new_protected:Nn \__enumext_after_list_vii:
4495 {
4496     \__enumext_stop_mini_vii:
4497     \__enumext_after_stop_list_vii:
4498     \__enumext_check_ans_key_hook:
4499     \__enumext_vspace_below_vii:
4500     \bool_set_false:N \l__enumext_starred_bool
4501     \__enumext_resume_save_counter:
4502 }

```

(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`.

```

4503 \cs_new_protected:Nn \__enumext_start_store_level_vii:
4504 {
4505     \bool_if:NT \l__enumext_store_active_bool
4506     {
4507         \int_compare:nNnT { \l__enumext_level_int } > { 0 }
4508         {
4509             \__enumext_store_level_open_vii:
4510         }
4511     }
4512 }
4513 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
4514 {
4515     \bool_if:NT \l__enumext_store_active_bool
4516     {
4517         \int_compare:nNnT { \l__enumext_level_int } > { 0 }
4518         {
4519             \__enumext_store_level_close_vii:
4520         }
4521     }
4522 }

```

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

12.43.1 The command \item in enumext*

__enumext_first_item_tmp_vii:

The __enumext_first_item_tmp_vii: function will remove horizontal space equal to `\labelwidth` plus `\labelsep` to the left of the first `\item` in the environment at the point of execution of this function, where it is equal to the __enumext_stop_item_tmp_vii: function inside the environment body definition.

```

4523 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_vii:
4524 {
4525     \skip_horizontal:n { -\l__enumext_labelwidth_vii_dim - \l__enumext_labelsep_vii_dim }
4526 }

```

(End of definition for __enumext_first_item_tmp_vii:.)

__enumext_start_item_tmp_vii:

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

```

4527 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
4528 {

```

```

4529     \__enumext_stop_item_tmp_vii:
4530     \int_incr:N \__enumext_item_column_pos_vii_int
4531     \int_gincr:N \g__enumext_item_count_all_vii_int
4532     \__enumext_item_peek_args_vii:
4533 }

```

(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).

```

4534 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
4535 {
4536   \peek_meaning:NTF (
4537     { \__enumext_joined_item_vii:w }
4538     { \__enumext_joined_item_vii:w (1) }
4539   }

```

(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.

```

4540 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
4541 {
4542   \__enumext_starred_joined_item_vii:n {#1}
4543   \peek_meaning_remove:NTF *
4544     { \__enumext_starred_item_vii:w }
4545     { \__enumext_standar_item_vii:w }
4546 }

```

(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].

```

4547 \cs_new_protected:Npn \__enumext_standar_item_vii:w
4548 {
4549   \bool_set_false:N \l__enumext_item_starred_vii_bool
4550   \peek_meaning:NTF [
4551     {
4552       \bool_set_eq:NN \l__enumext_wrap_label_vii_bool \l__enumext_wrap_label_opt_vii_bool
4553       \__enumext_start_item_vii:w
4554     }
4555     {
4556       \bool_set_true:N \l__enumext_wrap_label_vii_bool
4557       \legacy_if_set_true:n { @noitemarg }
4558       \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4559     }
4560   }

```

(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

```

4561 \cs_new_protected:Npn \__enumext_starred_item_vii:w
4562 {
4563   \bool_set_true:N \l__enumext_item_starred_vii_bool
4564   \bool_set_true:N \l__enumext_wrap_label_vii_bool
4565   \peek_meaning:NTF [
4566     { \__enumext_starred_item_vii_aux_i:w }
4567     { \__enumext_starred_item_vii_aux_ii:w }
4568   }
4569   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
4570   {

```

```

4571 \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
4572 \__enumext_starred_item_vii_aux_ii:w
4573 }
4574 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
4575 {
4576 \peek_meaning:NTF [
4577 { \__enumext_starred_item_vii_aux_iii:w }
4578 {
4579 \dim_set_eq:NN \l__enumext_item_symbol_sep_vii_dim \l__enumext_labelsep_vii_dim
4580 \legacy_if_set_true:n { @noitemarg }
4581 \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4582 }
4583 }
4584 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
4585 {
4586 \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
4587 \legacy_if_set_true:n { @noitemarg }
4588 \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4589 }

```

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

`__enumext_fake_make_label_vii:n`

The `__enumext_fake_make_label_vii:n` function will be in charge of handling our definition of `\item`. First we increment the counter `enumXvii` for the enumerated items and activate support for the *check answers* mechanism, followed by support for `\item*[\langle symbol \rangle][\langle offset \rangle]` if present, then the `wrap-label` and `wrap-label*` keys which we execute using `\makebox` whose width will be given by the `labelwidth` key and position by the `align` key, inside the argument of this we will execute the `font` key together with the function defined by the `wrap-label` or `wrap-label*` keys. Finally we execute the `labelsep` key applying a *horizontal space*.

```

4590 \cs_new_protected_nopar:Npn \__enumext_fake_make_label_vii:n #1
4591 {
4592 \legacy_if:nT { @noitemarg }
4593 {
4594 \legacy_if_set_false:n { @noitemarg }
4595 \legacy_if:nT { @nmbrrlist }
4596 {
4597 \refstepcounter{enumXvii}
4598 \bool_if:NT \l__enumext_check_answers_bool
4599 {
4600 \int_gincr:N \g__enumext_item_number_int
4601 \bool_set_true:N \l__enumext_item_number_bool
4602 }
4603 }
4604 }
4605 \bool_if:NT \l__enumext_item_starred_vii_bool
4606 {
4607 \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
4608 {
4609 \tl_gset_eq:NN
4610 \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
4611 }
4612 \mode_leave_vertical:
4613 \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
4614 \hbox_overlap_left:n { \g__enumext_item_symbol_aux_vii_tl }
4615 \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
4616 \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
4617 }
4618 \bool_if:NTF \l__enumext_wrap_label_vii_bool
4619 {
4620 \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
4621 {
4622 \tl_use:N \l__enumext_label_font_style_vii_tl
4623 \__enumext_wrapper_label_vii:n {#1}
4624 }
4625 }
4626 {
4627 \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
4628 {
4629 \tl_use:N \l__enumext_label_font_style_vii_tl #1
4630 }

```

```

4631     }
4632     \skip_horizontal:N \l__enumext_labelsep_vii_dim
4633 }

```

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

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

The functions `__enumext_start_item_vii:w` and `__enumext_stop_item_vii:` executing the true definition of `\item` inside the `enumext*` environment, unlike the implementation in `shortlst` we will NOT use an extra group and the plain form of the `lrbox` environment.

`__enumext_start_item_vii:w` The first thing we will do is set the value of `__enumext_stop_item_tmp_vii:` equal to `__enumext_stop_item_vii:` which we will define later, after that we will start capturing `\item` and its `⟨contents⟩` in a *horizontal box* where the width will be `\itemwidth` plus `\labelwidth` plus `\labelsep`.

```

4634 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
4635 {
4636   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
4637   \hbox_set_to_wd:Nnw \l__enumext_item_text_vii_box
4638   {
4639     \l__enumext_joined_width_vii_dim
4640     + \l__enumext_labelwidth_vii_dim
4641     + \l__enumext_labelsep_vii_dim
4642   }

```

If `\DocumentMetadata` is not active and the state of the variable `\l__enumext_footnotes_key_bool` is false, we will redefine the `\footnote` command.

```

4643   \IfDocumentMetadataTF { }
4644   {
4645     \bool_if:NF \l__enumext_footnotes_key_bool
4646     {
4647       \__enumext_renew_footnote:
4648     }
4649   }

```

Now we insert our *sockets* for the tagPDF support and print `\item`.

```

4650   \__enumext_start_list_tag:n {enumext*}
4651   \__enumext_fake_make_label_vii:n {#1}
4652   \__enumext_stop_start_list_tag:

```

Finally we open the `minipage` environment capture the `⟨item content⟩` and execute the `first key`, `listparindent` key which will be equal to `\parindent`, the `parsep` key which will be equal to `\parskip` and the `itemindent` key.

```

4653   \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
4654   \tl_use:N \l__enumext_after_list_args_vii_tl
4655   \dim_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
4656   \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
4657   \tl_use:N \l__enumext_fake_item_indent_vii_tl
4658 }

```

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

`__enumext_stop_item_vii:` The `__enumext_stop_item_vii:` function will finish the fetching `\item` and its `⟨content⟩` by closing the `minipage` environment, the *sockets* for the tagPDF and the *horizontal box*.

```

4659 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
4660 {
4661   \__enumext_endminipage:
4662   \__enumext_stop_list_tag:n {enumext*}
4663   \hbox_set_end:

```

Here we will reduce the *warnings* a bit by setting the value of `\hbadness` to `10000`, print the `⟨contents⟩` of the *box* along with `\footnote`.

```

4664   \int_set:Nn \hbadness { 10000 }
4665   \box_use_drop:N \l__enumext_item_text_vii_box
4666   \IfDocumentMetadataTF { }
4667   {
4668     \bool_if:NF \l__enumext_footnotes_key_bool
4669     {
4670       \__enumext_print_footnote:
4671     }
4672   }

```

Finally set the *vertical* and *horizontal* spaces between rows and columns.

```

4673 \int_compare:nNnTF
4674 { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
4675 {
4676   \par\noindent
4677   \int_zero:N \l__enumext_item_column_pos_vii_int
4678 }
4679 {
4680   \skip_horizontal:N \l__enumext_columns_sep_vii_dim
4681 }
4682 }

```

(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*.

```

4683 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
4684 {
4685   \int_compare:nNnT
4686   {
4687     \int_mod:nn
4688     { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
4689   }
4690   =
4691   { 0 }
4692   {
4693     \para_end:
4694     \skip_vertical:n { -\l__enumext_itemsep_vii_skip }
4695     \skip_vertical:N \c_zero_skip
4696     \int_gzero:N \g__enumext_item_count_all_vii_int
4697   }
4698 }

```

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`.

```

4699 \__enumext_after_env:nn {enumext*} { \__enumext_execute_after_env: }

```

(End of definition for `__enumext_remove_extra_parsep_vii:`)

12.44 The environment `keyans*`

keyans*

First we will generate the environment and we will give a temporary definition to `__enumext_stop_item_tmp_viii:` equal to `__enumext_first_item_tmp_viii:` and next to `\item` equal to `__enumext_start_item_tmp_viii:` which we will redefine later. The implementation of this environment is the same as that used by the `enumext*` environment except for the `__enumext_check_starred_cmd:n` function added in the second part.

```

4700 \NewDocumentEnvironment{keyans*}{ o }
4701 {
4702   \__enumext_safe_exec_viii:
4703   \__enumext_parse_keys_viii:n {#1}
4704   \__enumext_before_list_viii:
4705   \__enumext_start_list:nn { }
4706   {
4707     \__enumext_list_arg_two_viii:
4708     \__enumext_before_keys_exec_viii:
4709   }
4710   \IfDocumentMetadataTF { \tag_suspend:n {keyans*} } { } { }
4711   \__enumext_starred_columns_set_viii:
4712   \item[] \scan_stop:
4713   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_first_item_tmp_viii:
4714   \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
4715   \ignorespaces
4716 }
4717 {
4718   \IfDocumentMetadataTF { \tag_struct_end: } { } { }
4719   \__enumext_stop_item_tmp_viii:
4720   \__enumext_remove_extra_parsep_viii:
4721   \__enumext_check_starred_cmd:n { item }
4722   %%\__enumext_stop_list:

```

```

4723   \__enumext_after_list_viii:
4724   }

```

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

`__enumext_safe_exec_viii:`

The `__enumext_safe_exec_viii:` function will first check if the `save-ans` key is active and only when this is true the environment will be available, it will increment the value of `\l__enumext_keyans_level_h_int` and return an error message when we are nesting the environment, then it will call the `__enumext_keyans_name_and_start:` function in charge of saving the name of the environment and the line it is running on, then it will check if we are trying to nest `keyans*` in `enumext*` returning an error and we will set `\l__enumext_starred_bool` to true, finally we will check if we are within the appropriate level within the `enumext` environment.

```

4725 \cs_new_protected:Nn \__enumext_safe_exec_viii:
4726 {
4727   \bool_if:NF \l__enumext_store_active_bool
4728   {
4729     \msg_error:nnnn { enumext } { wrong-place } { keyans* } { save-ans }
4730   }
4731   \int_incr:N \l__enumext_keyans_level_h_int
4732   \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
4733   {
4734     \msg_error:nn { enumext } { nested }
4735   }
4736   \__enumext_keyans_name_and_start:
4737   \bool_if:NT \l__enumext_starred_bool
4738   {
4739     \msg_error:nnn { enumext } { nested-horizontal } { enumext* }
4740   }
4741   \bool_set_true:N \l__enumext_starred_bool
4742   % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
4743   \bool_set_false:N \l__enumext_store_active_bool
4744   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
4745   {
4746     \msg_error:nn { enumext } { keyans-wrong-level }
4747   }
4748 }

```

(End of definition for `__enumext_safe_exec_viii:`.)

`__enumext_parse_keys_viii:n`

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

```

4749 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
4750 {
4751   \tl_if_novalue:nF {#1}
4752   {
4753     \keys_set:nn { enumext / keyans* } {#1}
4754   }
4755 }

```

(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`.

```

4756 \cs_new_protected:Nn \__enumext_before_list_viii:
4757 {
4758   \__enumext_vspace_above_viii:
4759   \__enumext_before_args_exec_viii:
4760   \__enumext_start_mini_viii:
4761 }

```

(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.

```

4762 \cs_new_protected:Nn \__enumext_after_list_viii:
4763 {
4764   \__enumext_stop_mini_viii:
4765   \__enumext_after_stop_list_viii:
4766   \__enumext_vspace_below_viii:
4767 }

```

(End of definition for `__enumext_after_list_viii:`.)

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

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

`_enumext_first_item_tmp_viii:` The `_enumext_first_item_tmp_viii:` function will remove horizontal space equal to `\labelwidth` plus `\labelsep` to the left of the first `\item` in the environment at the point of execution of this function, where it is equal to the `_enumext_stop_item_tmp_viii:` function inside the environment body definition.

```

4768 \cs_new_protected_nopar:Nn \_enumext_first_item_tmp_viii:
4769 {
4770   \skip_horizontal:n { -\labelwidth_viii_dim - \labelsep_viii_dim }
4771 }

```

(End of definition for `_enumext_first_item_tmp_viii:`)

`_enumext_start_item_tmp_viii:` First we will call the function `_enumext_stop_item_tmp_viii:` that we will redefine later, we will increment the value of `\l_enumext_item_column_pos_viii_int` that will count the item's by rows and the value of `\g_enumext_item_count_all_viii_int` that will count the total of item's in the environment. After that we will call the function `_enumext_item_peek_args_viii:` that will handle the arguments passed to `\item`.

```

4772 \cs_new_protected_nopar:Nn \_enumext_start_item_tmp_viii:
4773 {
4774   \_enumext_stop_item_tmp_viii:
4775   \int_incr:N \l_enumext_item_column_pos_viii_int
4776   \int_gincr:N \g_enumext_item_count_all_viii_int
4777   \_enumext_item_peek_args_viii:
4778 }

```

(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).

```

4779 \cs_new_protected:Nn \_enumext_item_peek_args_viii:
4780 {
4781   \peek_meaning:NTF (
4782     { \_enumext_joined_item_viii:w }
4783     { \_enumext_joined_item_viii:w (1) }
4784   }

```

(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`.

```

4785 \cs_new_protected:Npn \_enumext_joined_item_viii:w (#1)
4786 {
4787   \_enumext_starred_joined_item_viii:n {#1}
4788   \peek_meaning_remove:NTF *
4789   { \_enumext_starred_item_viii:w }
4790   { \_enumext_standar_item_viii:w }
4791 }

```

(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]`.

```

4792 \cs_new_protected:Npn \_enumext_standar_item_viii:w
4793 {
4794   \bool_set_false:N \l_enumext_item_starred_viii_bool
4795   \peek_meaning:NTF [

```

```

4796     {
4797         \bool_set_eq:NN \l__enumext_wrap_label_viii_bool \l__enumext_wrap_label_opt_viii_bool
4798         \__enumext_start_item_viii:w
4799     }
4800     {
4801         \bool_set_true:N \l__enumext_wrap_label_viii_bool
4802         \legacy_if_set_true:n { @noitemarg }
4803         \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4804     }
4805 }

```

(End of definition for __enumext_standar_item_viii:w.)

```

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

```

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

```

4806 \cs_new_protected:Npn \__enumext_starred_item_viii:w
4807 {
4808     \bool_set_true:N \l__enumext_item_starred_viii_bool
4809     \bool_set_true:N \l__enumext_wrap_label_viii_bool
4810     \peek_meaning:NTF [
4811     { \__enumext_starred_item_viii_aux_i:w }
4812     { \__enumext_starred_item_viii_aux_ii:w }
4813 }

```

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.

```

4814 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
4815 {
4816     \tl_clear:N \l__enumext_store_current_label_tl
4817     \tl_if_no_value:nF { #1 }
4818     {
4819         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
4820         {
4821             \tl_put_right:Ne \l__enumext_store_current_label_tl
4822             {
4823                 \l__enumext_store_keyans_item_opt_sep_tl
4824             }
4825             \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
4826         }
4827         \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
4828     }
4829     \__enumext_starred_item_viii_aux_ii:w
4830 }
4831 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
4832 {
4833     \legacy_if_set_true:n { @noitemarg }
4834     \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4835 }

```

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

```
\__enumext_starred_item_exec:
```

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

```

4836 \cs_new_protected:Nn \__enumext_starred_item_exec:
4837 {
4838     \tl_put_left:Ne \l__enumext_store_current_label_tl { \l__enumext_label_viii_tl }
4839     \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
4840     \__enumext_keyans_store_ref:
4841     \tl_put_left:Ne \l__enumext_store_current_label_tl { \item }
4842     \__enumext_keyans_addto_seq_link:
4843     \int_gincr:N \g__enumext_check_starred_cmd_int
4844     \bool_if:NT \l__enumext_show_answer_bool
4845     {
4846         \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4847     }
4848     \bool_if:NT \l__enumext_show_position_bool
4849     {

```

```

4850         \tl_set:Nc \l__enumext_mark_answer_sym_tl
4851         {
4852             \group_begin:
4853             \exp_not:N \normalfont
4854             \exp_not:N \footnotesize [ \int_eval:n
4855             {
4856                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
4857             }
4858             ]
4859             \group_end:
4860         }
4861     \l__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4862 }
4863 }

```

(End of definition for \l__enumext_starred_item_exec:.)

\l__enumext_fake_make_label_viii:n

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

```

4864 \cs_new_protected_nopar:Npn \l__enumext_fake_make_label_viii:n #1
4865 {
4866     \legacy_if:nT { @noitemarg }
4867     {
4868         \legacy_if_set_false:n { @noitemarg }
4869         \legacy_if:nT { @nmbrrlist }
4870         {
4871             \refstepcounter{enumXviii}
4872         }
4873     }
4874     \bool_if:NT \l__enumext_item_starred_viii_bool
4875     {
4876         \l__enumext_starred_item_exec:
4877     }
4878     \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
4879     {
4880         \tl_use:N \l__enumext_label_font_style_viii_tl
4881         \bool_if:NTF \l__enumext_wrap_label_viii_bool
4882         {
4883             \l__enumext_wrapper_label_viii:n {#1}
4884         }
4885         { #1 }
4886     }
4887     \skip_horizontal:N \l__enumext_labelsep_viii_dim
4888 }

```

(End of definition for \l__enumext_fake_make_label_viii:n.)

12.44.2 Real definition of \item in keyans*

\l__enumext_start_item_viii:w

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

```

4889 \cs_new_protected_nopar:Npn \l__enumext_start_item_viii:w [#1]
4890 {
4891     \cs_set_eq:NN \l__enumext_stop_item_tmp_viii: \l__enumext_stop_item_viii:
4892     \hbox_set_to_wd:Nnw \l__enumext_item_text_viii_box
4893     {
4894         \l__enumext_joined_width_viii_dim
4895         + \l__enumext_labelwidth_viii_dim
4896         + \l__enumext_labelsep_viii_dim
4897     }
4898     \IfDocumentMetadataTF { }
4899     {
4900         \bool_if:NF \l__enumext_footnotes_key_bool
4901         {
4902             \l__enumext_renew_footnote:
4903         }
4904     }
4905     \l__enumext_start_list_tag:n {keyans*}
4906     \l__enumext_fake_make_label_viii:n {#1}
4907     \l__enumext_stop_start_list_tag:
4908     \l__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
4909     \tl_use:N \l__enumext_after_list_args_viii_tl
4910     \dim_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
4911     \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip

```

```

4912         \bool_if:NT \l__enumext_item_starred_viii_bool
4913         {
4914             \tl_use:N \l__enumext_fake_item_indent_viii_tl
4915             \__enumext_keyans_show_item_opt:
4916             \skip_horizontal:n { -\l__enumext_fake_item_indent_viii_dim - \l__enumext_labelsep_viii_dim }
4917         }
4918         {
4919             \tl_use:N \l__enumext_fake_item_indent_viii_tl
4920         }
4921     }

```

(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.

```

4922 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
4923 {
4924     \__enumext_endminipage:
4925     \__enumext_stop_list_tag:n {keyans*}
4926     \hbox_set_end:
4927     \int_set:Nn \hbadness { 10000 }
4928     \box_use_drop:N \l__enumext_item_text_viii_box
4929     \IfDocumentMetadataTF { }
4930     {
4931         \bool_if:NF \l__enumext_footnotes_key_bool
4932         {
4933             \__enumext_print_footnote:
4934         }
4935     }
4936     \int_compare:nNnTF
4937     { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
4938     {
4939         \par\noindent
4940         \int_zero:N \l__enumext_item_column_pos_viii_int
4941     }
4942     {
4943         \skip_horizontal:N \l__enumext_columns_sep_viii_dim
4944     }
4945 }

```

(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.

```

4946 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
4947 {
4948     \int_compare:nNnTF
4949     {
4950         \int_mod:nn
4951         { \g__enumext_item_count_all_viii_int }
4952         { \l__enumext_columns_viii_int }
4953     }
4954     =
4955     { 0 }
4956     {
4957         \para_end:
4958         \skip_vertical:n { -\l__enumext_itemsep_viii_skip }
4959         \skip_vertical:N \c_zero_skip
4960         \int_gzero:N \g__enumext_item_count_all_viii_int
4961     }
4962 }

```

(End of definition for __enumext_remove_extra_parsep_viii:.)

12.45 The command \getkeyans

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

```

4963 \NewDocumentCommand \getkeyans { m }
4964 {
4965   \exp_args:Ne \__enumext_getkeyans_aux:n
4966   { \tl_to_str:e { \text_expand:n {#1} } }
4967 }

```

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

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

```

4968 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
4969 {
4970   \str_if_in:nnTF {#1} { : }
4971   {
4972     \use:e
4973     {
4974       \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
4975       { {##1} {##2} }
4976     }
4977     \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
4978   }
4979   { \msg_error:nnn { enumext } { missing-colon } {#1} }
4980 }

```

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

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

```

4981 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
4982 {
4983   \prop_if_exist:cTF { g__enumext_#1_prop }
4984   {
4985     \prop_item:cn { g__enumext_#1_prop } {#2}
4986   }
4987   {
4988     \msg_error:nnn { enumext } { undefined-storage-anskey } {#1}
4989   }
4990 }

```

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

12.46 The command \printkeyans

The `\printkeyans` command prints “all stored content” in the $\langle sequence \rangle$ defined by the `save-ans` key. The first thing we will do is define a set of $\langle filtered\ keys \rangle$ with which we will control the options of the different nesting levels for the environment `enumext` and `enumext*` by storing their values in the list of tokens `\l__enumext_print_keyans_X_tl`.

The variable `\l__enumext_print_keyans_starred_tl` will have the default $\langle keys \rangle$ for `\printkeyans*` and will be set by `\setenumext[$\langle print^* \rangle$]` and the variable `\l__enumext_print_keyans_vii_tl` will have the default keys for the environment `enumext*` nested within the $\langle sequence \rangle$ and will be set by `\setenumext[$\langle print, * \rangle$]`, the rest of the variables will be for the environment `enumext` and will be set by `\setenumext[$\langle print, level \rangle$]`.

```

4991 \keys_define:nn { enumext / print }
4992 {
4993   print* .code:n = \keys_precompile:neN { enumext / enumext* }
4994               { \__enumext_filter_save_key:n {#1} }
4995               \l__enumext_print_keyans_starred_tl, % starred cmd
4996   print* .initial:n = { nosep, label=\arabic*, columns=2, first=\small, font=\small },
4997   print-1 .code:n = \keys_precompile:neN { enumext / level-1 }
4998               { \__enumext_filter_save_key:n {#1} }
4999               \l__enumext_print_keyans_i_tl,
5000   print-1 .initial:n = { nosep, label=\arabic*, columns=2, first=\small, font=\small },
5001   print-2 .code:n = \keys_precompile:neN { enumext / level-2 }
5002               { \__enumext_filter_save_key:n {#1} }
5003               \l__enumext_print_keyans_ii_tl,
5004   print-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },

```

```

5005     print-3 .code:n      = \keys_precompile:neN { enumext / level-3 }
5006                           { \__enumext_filter_save_key:n {#1} }
5007                           \l__enumext_print_keyans_iii_tl,
5008     print-3 .initial:n   = { nosep, label=\roman*., first=\small, font=\small },
5009     print-4 .code:n      = \keys_precompile:neN { enumext / level-4 }
5010                           { \__enumext_filter_save_key:n {#1} }
5011                           \l__enumext_print_keyans_iv_tl,
5012     print-4 .initial:n   = { nosep, label=\Alph*., first=\small, font=\small },
5013     print-* .code:n      = \keys_precompile:neN { enumext / enumext* }
5014                           { \__enumext_filter_save_key:n {#1} }
5015                           \l__enumext_print_keyans_vii_tl, % starred nested
5016     print-* .initial:n   = { nosep, label=\arabic*., first=\small, font=\small },
5017   }

```

• 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`.

```

5018 \NewDocumentCommand \printkeyans { s O{} m }
5019 {
5020   \group_begin:
5021     \tl_use:N \l__enumext_print_keyans_i_tl
5022     \tl_use:N \l__enumext_print_keyans_ii_tl
5023     \tl_use:N \l__enumext_print_keyans_iii_tl
5024     \tl_use:N \l__enumext_print_keyans_iv_tl
5025     \tl_use:N \l__enumext_print_keyans_vii_tl
5026     \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
5027   \group_end:
5028 }

```

(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.

```

5029 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
5030 {
5031   \seq_if_exist:cTF { g__enumext_#3_seq }
5032   {
5033     \seq_if_empty:cF { g__enumext_#3_seq }
5034     {
5035       %%\seq_show:c { g__enumext_#3_seq }

```

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

```

5036       \bool_if:nTF {#1}
5037       {
5038         \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
5039         {
5040           \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
5041         }
5042         {
5043           \tl_use:N \l__enumext_print_keyans_starred_tl
5044           \begin{enumext*}[#2]
5045             \keys_set:nn { enumext / level-1 }{ base-fix }
5046             \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5047             \end{enumext*}
5048           }
5049       }

```

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

```

5050       {
5051         \begin{enumext}[#2]
5052         \keys_set:nn { enumext / enumext* }{ base-fix }
5053         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5054         \end{enumext}

```

```

5055         }
5056     }
5057 }
5058 {
5059     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
5060 }
5061 }

```

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

12.47 The command `\setenumext`

The command `\setenumext` will be in charge of managing the *⟨keys⟩* passed to all environments and to the `\printkeyans` command. We must take precautions with the `enumext*` environment and “first level” of the `enumext` environment so as not to capture *⟨keys⟩* that complicate us.

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

```

\__enumext_filter_first_level:n
\__enumext_filter_first_level_key:n
\__enumext_filter_first_level_pair:nn
5062 \cs_new:Npn \__enumext_filter_first_level:n #1
5063 {
5064     \use:e
5065     {
5066         \keyval_parse:NNn
5067         \__enumext_filter_first_level_key:n
5068         \__enumext_filter_first_level_pair:nn {#1}
5069     }
5070 }

```

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

```

5071 \cs_new:Npn \__enumext_filter_first_level_key:n #1
5072 {
5073     \str_case:nnF {#1}
5074     {
5075         { resume } {}
5076         { resume* } {}
5077     }
5078     { , { \exp_not:n {#1} } } }
5079 }

```

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

```

5080 \cs_new:Npn \__enumext_filter_first_level_pair:nn #1#2
5081 {
5082     \str_case:nnF {#1}
5083     {
5084         { series } {}
5085         { resume } {}
5086         { save-ans } {}
5087     }
5088     { , { \exp_not:n {#1} } = { \exp_not:n {#2} } } }
5089 }

```

(End of definition for `__enumext_filter_first_level:n`, `__enumext_filter_first_level_key:n`, and `__enumext_filter_first_level_pair:nn`.)

Now define a “meta families” of *⟨keys⟩* to access from `\setenumext`.

```

5090 \keys_define:nn { enumext / meta-families }
5091 {
5092     enumext-1 .code:n =
5093     {
5094         \keys_set:ne { enumext / level-1 }
5095         {
5096             \__enumext_filter_first_level:n {#1}
5097         }
5098     } ,
5099     enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
5100     enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
5101     enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
5102     keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} } ,
5103     enumext* .code:n =
5104     {
5105         \keys_set:ne { enumext / enumext* }

```



```

5106         {
5107             \__enumext_filter_first_level:n {#1}
5108         }
5109     } ,
5110     keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
5111     print* .code:n = { \keys_set:nn { enumext / print } { print* = {#1} } } ,
5112     print-1 .code:n = { \keys_set:nn { enumext / print } { print-1 = {#1} } } ,
5113     print-2 .code:n = { \keys_set:nn { enumext / print } { print-2 = {#1} } } ,
5114     print-3 .code:n = { \keys_set:nn { enumext / print } { print-3 = {#1} } } ,
5115     print-4 .code:n = { \keys_set:nn { enumext / print } { print-4 = {#1} } } ,
5116     print-* .code:n = { \keys_set:nn { enumext / print } { print-* = {#1} } } ,
5117     unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
5118 }

```

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

```

5119 \seq_const_from_clist:Nn \__enumext_all_families_seq
5120 {
5121     enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
5122     keyans*, print-1, print-2, print-3, print-4, print-*, print*,
5123 }

```

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

```

5124 \NewDocumentCommand \setenumext { 0{enumext,1} +m }
5125 {
5126     \seq_clear:N \__enumext_setkey_tmpa_seq
5127     \seq_set_from_clist:Nn \__enumext_setkey_tmpb_seq {#1}
5128     \int_set:Nn \__enumext_setkey_tmpa_int
5129     {
5130         \seq_count:N \__enumext_setkey_tmpb_seq
5131     }
5132     \int_compare:nNnTF { \__enumext_setkey_tmpa_int } > { 1 }
5133     {
5134         \seq_pop_left:NN \__enumext_setkey_tmpb_seq \__enumext_setkey_tmpa_tl
5135         \seq_map_function:NN \__enumext_setkey_tmpb_seq \__enumext_set_parse:n
5136         \seq_set_map_e:NNn \__enumext_setkey_tmpa_seq \__enumext_setkey_tmpa_seq
5137         {
5138             \tl_use:N \__enumext_setkey_tmpa_tl - ##1
5139         }
5140     }
5141     {
5142         \seq_put_right:Ne \__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
5143     }
5144     \seq_if_empty:NNTF \__enumext_setkey_tmpa_seq
5145     { \seq_map_inline:Nn \__enumext_all_families_seq }
5146     { \seq_map_inline:Nn \__enumext_setkey_tmpa_seq }
5147     {
5148         \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
5149     }
5150 }

```

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

`__enumext_set_parse:n`
`__enumext_set_error:nn`

Internal functions used by the `\setenumext` command.

```

5151 \cs_new_protected:Npn \__enumext_set_parse:n #1
5152 {
5153     \tl_set:Ne \__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
5154     \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
5155     { \tl_remove_all:Nn \__enumext_setkey_tmpb_tl {##1} }
5156     \tl_if_empty:NNTF \__enumext_setkey_tmpb_tl
5157     {
5158         \seq_put_right:Ne \__enumext_setkey_tmpa_seq
5159         { \tl_trim_spaces:n {#1} }
5160     }
5161     { \__enumext_set_error:nn {#1} { } }
5162 }
5163 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
5164 { \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

5165 \prop_const_from_keyval:Nn \c__enumext_meta_paths_prop
5166 {
5167   {enumext,1} = level-1,
5168   {enumext,2} = level-2,
5169   {enumext,3} = level-3,
5170   {enumext,4} = level-4,
5171   {enumext*} = enumext*
5172 }
```

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

```
5173 \NewDocumentCommand \setenumextmeta { s O{enumext,1} m +m }
5174 {
5175   \str_if_eq:eeTF { \tl_trim_spaces:n {#3} } { unknown }
5176   { \msg_error:nn { enumext } { prohibited-unknown } }
5177   {
5178     \bool_if:nTF {#1}
5179     {
5180       \int_step_inline:nn { 4 }
5181       { __enumext_add_meta_key:nnn { enumext, ##1 } {#3} {#4} }
5182       __enumext_add_meta_key:nnn { enumext* } {#3} {#4}
5183     }
5184     { __enumext_add_meta_key:nnn {#2} {#3} {#4} }
5185   }
5186 }
```

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

```
5187 \cs_new_protected:Npn __enumext_add_meta_key:nnn #1
5188 {
5189   \tl_set:Nn \l__enumext_meta_path_tl {#1}
5190   \tl_replace_all:Nnn \l__enumext_meta_path_tl { ~ } {}
5191   \prop_get:NVNTF
5192   \c__enumext_meta_paths_prop \l__enumext_meta_path_tl \l__enumext_meta_path_tl
5193   { __enumext_def_meta_key:Vnn \l__enumext_meta_path_tl }
5194   {
5195     \msg_error:nnn { enumext } { unknown-set } {#1}
5196     \use_none:n
5197   }
5198 }
5199 \cs_new_protected:Npn __enumext_def_meta_key:nnn #1#2#3
5200 {
5201   \bool_lazy_or:nnTF
5202   { \keys_if_exist_p:nn { enumext / #1 } {#2} }
5203   { \keys_if_exist_p:nn { enumext / enumext* } {#2} }
5204   { \msg_error:nnn { enumext } { already-defined } {#2} }
5205   {
5206     \keys_define:nn { enumext / #1 }
5207     {
5208       #2 .meta:n = {#3},
5209       #2 .value_forbidden:n = true
5210     }
5211   }
5212 }
5213 \cs_generate_variant:Nn __enumext_def_meta_key:nnn { V }
```

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

12.49 The command \foreachkeyans

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

`\foreachkeyans`

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

```
\enumext_parse_foreach_keys:nn
__enumext_parse_foreach_keys:nn
5214 \keys_define:nn { enumext / foreach }
```

©2024 by Pablo González L

```

5215 {
5216     before .tl_set:N = \l__enumext_foreach_before_tl,
5217     before .value_required:n = true,
5218     after .tl_set:N = \l__enumext_foreach_after_tl,
5219     after .value_required:n = true,
5220     start .int_set:N = \l__enumext_foreach_start_int,
5221     start .value_required:n = true,
5222     stop .int_set:N = \l__enumext_foreach_stop_int,
5223     stop .value_required:n = true,
5224     step .int_set:N = \l__enumext_foreach_step_int,
5225     step .value_required:n = true,
5226     wrapper .cs_set_protected:Np = \l__enumext_foreach_wrapper:n #1,
5227     wrapper .value_required:n = true,
5228     sep .tl_set:N = \l__enumext_foreach_sep_tl,
5229     sep .value_required:n = true,
5230     unknown .code:n = { \l__enumext_parse_foreach_keys:n {#1} }
5231 }
5232 \keys_precompile:nnN { enumext / foreach }
5233 {
5234     before={},after={},start=1,step=1,stop=0,wrapper=#1,sep=
5235 }
5236 \g__enumext_foreach_default_keys_tl

```

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

```

5237 \cs_new_protected:Npn \l__enumext_parse_foreach_keys:nn #1#2
5238 {
5239     \tl_if_blank:nTF {#2}
5240     {
5241         \msg_error:nnn { enumext } { for-key-unknown } {#1}
5242     }
5243     {
5244         \msg_error:nnnn { enumext } { for-key-value-unknown } {#1} {#2}
5245     }
5246 }
5247 \cs_new_protected:Npn \l__enumext_parse_foreach_keys:n #1
5248 {
5249     \exp_args:NV \l__enumext_parse_foreach_keys:nn \l_keys_key_str {#1}
5250 }

```

We create the command.

```

5251 \NewDocumentCommand \foreachkeyans { +0{} m }
5252 {
5253     \l__enumext_foreach_keyans:nn {#1} {#2}
5254 }

```

Finally the internal functions $\l__enumext_foreach_keyans:nn$ and $\l__enumext_foreach_add_body:n$ will loop through the prop list and print the contents.

```

5255 \cs_new_protected:Npn \l__enumext_foreach_keyans:nn #1 #2
5256 {
5257     \tl_use:N \g__enumext_foreach_default_keys_tl
5258     \keys_set:nn { enumext / foreach } {#1}
5259     \tl_set:Nn \l__enumext_foreach_name_prop_tl {#2}
5260     \prop_if_exist:cF { g__enumext_#2_prop }
5261     {
5262         \msg_error:nnn { enumext } { undefined-storage-anskey } {#2}
5263     }
5264     \int_compare:nNnT { \l__enumext_foreach_stop_int } = { 0 }
5265     {
5266         \int_set:Nn \l__enumext_foreach_stop_int
5267         { \prop_count:c { g__enumext_#2_prop } }
5268     }
5269     \seq_clear:N \l__enumext_foreach_print_seq
5270     \int_step_function:nnnN
5271     { \l__enumext_foreach_start_int }
5272     { \l__enumext_foreach_step_int }
5273     { \l__enumext_foreach_stop_int }
5274     \l__enumext_foreach_add_body:n
5275     \seq_use:NV \l__enumext_foreach_print_seq \l__enumext_foreach_sep_tl
5276 }
5277 \cs_new_protected:Npn \l__enumext_foreach_add_body:n #1
5278 {
5279     \seq_put_right:Ne \l__enumext_foreach_print_seq

```

```

5280     {
5281         \exp_not:V \l__enumext_foreach_before_tl
5282         \__enumext_foreach_wrapper:n
5283         {
5284             \prop_item:cn { g__enumext_ \l__enumext_foreach_name_prop_tl _prop }{#1}
5285         }
5286         \exp_not:V \l__enumext_foreach_after_tl
5287     }
5288 }

```

(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.

```

5289 \msg_new:nnn { enumext } { package-load }
5290 {
5291     The ~ '#1' ~ package ~ is ~ already ~ loaded.
5292 }
5293 \msg_new:nnn { enumext } { package-not-load }
5294 {
5295     The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
5296 }
5297 \msg_new:nnn { enumext } { package-load-foot }
5298 {
5299     The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
5300 }

```

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

```

5301 \msg_new:nnn { enumext } { counters }
5302 {
5303     The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
5304     package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
5305 }

```

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

```

5306 \msg_new:nnn { enumext } { unknown-choice }
5307 {
5308     The ~ value ~ '#3' ~ for ~ '#1' ~ key ~ is ~ invalid ~ use ~ ('#2').
5309 }

```

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

```

5310 \msg_new:nnnn { enumext } { anskey-env-error }
5311 {
5312     The ~ '#1' ~ environment ~is~ ~ reserved ~ by ~\\
5313     'enumext' ~ package, ~ It~ is~ already~ defined.
5314 }
5315 {
5316     The ~ anskey* ~ environment ~ is ~ defined ~ internally ~
5317     for ~ the ~ 'save-ans' ~ key.\\
5318 }

```

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

```

5319 \msg_new:nnn { enumext } { store-prop }
5320 {
5321     * ~ Package ~ enumext: ~ Creating ~
5322     \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
5323 }
5324 \msg_new:nnn { enumext } { store-seq }
5325 {
5326     * ~ Package ~ enumext: ~ Creating ~
5327     \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
5328 }
5329 \msg_new:nnn { enumext } { store-int }
5330 {
5331     * ~ Package ~ enumext: ~ Creating ~
5332     \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_context:.
5333 }
5334 \msg_new:nnn { enumext } { prop-seq-int-hook }
5335 {
5336     * ~ Package ~ enumext: ~ Elements ~ in ~
5337     \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
5338     * ~ Package ~ enumext: ~ Elements ~ in ~

```

```

5339 \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
5340 * ~ Package ~ enumext: ~ Value ~ off ~
5341 \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
5342 }
5343 \msg_new:nnn { enumext } { item-answer-hook }
5344 {
5345   * ~ Package ~ enumext: ~ Value ~ off ~
5346   \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\\
5347   * ~ Package ~ enumext: ~ Value ~ off ~
5348   \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\\
5349   * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
5350 }

```

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

```

5351 \msg_new:nnn { enumext } { invalid-key }
5352 {
5353   The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
5354 }
5355 \msg_new:nnn { enumext } { unknown-key-family }
5356 {
5357   Unknown~key~family~`\l_keys_key_str'~for~enumext.
5358 }

```

Messages used in length calculation.

```

5359 \msg_new:nnn { enumext } { width-negative }
5360 {
5361   Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
5362   The ~ key ~ '#1'~ accepts ~ values ~ >= ~ 0pt.
5363 }
5364 \msg_new:nnn { enumext } { width-zero }
5365 {
5366   Invalid ~ '#1=#2' ~ \msg_line_context:.\
5367   The ~ key ~ '#1'~ accepts ~ values ~ > ~ 0pt.
5368 }

```

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

```

5369 \msg_new:nnn { enumext } { list-lengths }
5370 {
5371   **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
5372   \__enumext_show_length:nnn { dim } { labelsep } {#1}
5373   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
5374   \__enumext_show_length:nnn { dim } { itemindent } {#1}
5375   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
5376   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
5377   \__enumext_show_length:nnn { dim } { listparindent } {#1}
5378   \__enumext_show_length:nnn { skip } { topsep } {#1}
5379   \__enumext_show_length:nnn { skip } { parsep } {#1}
5380   \__enumext_show_length:nnn { skip } { partopsep } {#1}
5381   \__enumext_show_length:nnn { skip } { itemsep } {#1}
5382   ****
5383 }

```

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

```

5384 \msg_new:nnn { enumext } { list-lengths-not-nested }
5385 {
5386   **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
5387   \__enumext_show_length:nnn { dim } { labelsep } {#1}
5388   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
5389   \__enumext_show_length:nnn { dim } { itemindent } {#1}
5390   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
5391   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
5392   \__enumext_show_length:nnn { dim } { listparindent } {#1}
5393   \__enumext_show_length:nnn { skip } { topsep } {#1}
5394   \__enumext_show_length:nnn { skip } { parsep } {#1}
5395   \__enumext_show_length:nnn { skip } { partopsep } {#1}
5396   \__enumext_show_length:nnn { skip } { itemsep } {#1}
5397   ****
5398 }

```

Messages used by `ref` key.

```

5399 \msg_new:nnn { enumext } { key-ref-empty }
5400 {
5401   Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
5402 }

```

Messages used by `save-ans` key.

```

5403 \msg_new:nnn { enumext } { save-ans-empty }
5404 {
5405   Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1' ~ \msg_line_context:.
5406 }
5407 \msg_new:nnn { enumext } { save-ans-log }
5408 {
5409   * ~ Package ~ enumext: ~ Start ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5410 }
5411 \msg_new:nnn { enumext } { save-ans-log-hook }
5412 {
5413   * ~ Package ~ enumext: ~ Stop ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5414 }
5415 \msg_new:nnn { enumext } { save-ans-hook }
5416 {
5417   Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
5418 }

```

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

```

5419 \msg_new:nnn { enumext } { need-save-ans }
5420 {
5421   Key ~ '#1' ~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2' ~ \msg_line_context:.
5422 }
5423 \msg_new:nnn { enumext } { items-same-answer }
5424 {
5425   *****\
5426   * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~
5427   for ~ \c_left_brace_str #2 \c_right_brace_str\
5428   * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~
5429   'OK', ~ all ~ items ~ with ~ answer.\
5430   *****
5431 }
5432 \msg_new:nnn { enumext } { item-greater-answer }
5433 {
5434   Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\
5435   started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\
5436   Items ~ > ~ Answers.
5437 }
5438 \msg_new:nnn { enumext } { item-less-answer }
5439 {
5440   Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\
5441   started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\
5442   Items ~ < ~ Answers.
5443 }

```

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

```

5444 \msg_new:nnn { enumext } { missing-starred }
5445 {
5446   Missing ~ '\c_backslash_str #1*' ~ #2.
5447 }
5448 \msg_new:nnn { enumext } { many-starred }
5449 {
5450   Many ~ '\c_backslash_str #1*' ~ #2.
5451 }

```

Messages used by `\printkeyans*` command.

```

5452 \msg_new:nnn { enumext } { print-starred }
5453 {
5454   \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
5455   #2 ~ environment ~ \msg_line_context:.
5456 }

```

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

```

5457 \msg_new:nnn { enumext } { list-too-deep }
5458 {
5459   Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \
5460   The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
5461 }

```

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

```

5462 \msg_new:nnn { enumext } { anskey-unnumber-item }
5463 {
5464   Can't ~ store ~ with ~ a ~ unnumbered ~ \c_backslash_str item ~ \msg_line_context:.

```

```

5465 }
5466 \msg_new:nnn { enumext } { anskey-already-stored }
5467 {
5468   Content ~ already ~ stored ~ for ~ this ~ \c_backslash_str item ~ \msg_line_context:.
5469 }
5470 \msg_new:nnn { enumext } { anskey-empty-arg }
5471 {
5472   Can't ~ store ~ empty ~ content ~ \msg_line_context:.
5473 }
5474 \msg_new:nnn { enumext } { anskey-wrong-place }
5475 {
5476   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5477   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5478 }
5479 \msg_new:nnn { enumext } { anskey-nested }
5480 {
5481   The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
5482 }
5483 \msg_new:nnn { enumext } { anskey-math-mode }
5484 {
5485   #1 ~ can't ~ work ~ in ~ math ~ mode ~ \msg_line_context:.
5486 }
5487 \msg_new:nnn { enumext } { anskey-env-wrong }
5488 {
5489   The ~ environment ~ anskey* ~ cannot ~ use ~ in ~ '#1' ~ \msg_line_context:.
5490 }
5491 \msg_new:nnn { enumext } { ansPIC-wrong-place }
5492 {
5493   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5494   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5495 }
5496 \msg_new:nnn { enumext } { command-wrong-place }
5497 {
5498   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5499   '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
5500 }
5501 \msg_new:nnnn { enumext } { anskey-env-key-unknown }
5502 {
5503   The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5504   'anskey*' ~ and ~ is ~ being ~ ignored.
5505 }
5506 {
5507   The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5508   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5509 }
5510 \msg_new:nnnn { enumext } { anskey-env-key-value-unknown }
5511 {
5512   The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5513   'anskey*' ~ and ~ is ~ being ~ ignored.
5514 }
5515 {
5516   The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5517   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5518 }
5519 \msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
5520 { The ~ key ~ '#1' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.}
5521 {
5522   The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5523   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5524 }
5525 \msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
5526 { The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.}
5527 {
5528   The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5529   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5530 }

```

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

```

5531 \msg_new:nnn { enumext } { keyans-nested }
5532 {
5533   The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
5534 }

```



```

5535 \msg_new:nnn { enumext } { keyans-wrong-level }
5536 {
5537   Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
5538   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5539 }
5540 \msg_new:nnn { enumext } { wrong-place }
5541 {
5542   Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:~ \\
5543   '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext'.
5544 }
5545 \msg_new:nnn { enumext } { keyanspic-nested }
5546 {
5547   The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested~ \msg_line_context:~.
5548 }
5549 \msg_new:nnn { enumext } { keyanspic-wrong-level }
5550 {
5551   Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
5552   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5553 }
5554 \msg_new:nnn { enumext } { keyanspic-item-cmd }
5555 {
5556   Can't ~ use ~ \c_backslash_str item ~ in ~ keyanspic ~ \msg_line_context:.
5557 }
5558 \msg_new:nnnn { enumext } { keyans-unknown-key }
5559 {
5560   The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5561   '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5562 }
5563 {
5564   The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5565   ~ have ~ a ~ key ~ called ~ '#1'.\\
5566   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5567 }
5568 \msg_new:nnnn { enumext } { keyans-unknown-key-value }
5569 {
5570   The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5571   '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5572 }
5573 {
5574   The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5575   ~ have ~ a ~ key ~ called ~ '#1'.\\
5576   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5577 }

```

Message used by unknown $\langle keys \rangle$ in `enumext*`. environment.

```

5578 \msg_new:nnnn { enumext } { starred-unknown-key }
5579 {
5580   The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5581   '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5582 }
5583 {
5584   The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5585   ~ have ~ a ~ key ~ called ~ '#1'.\\
5586   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5587 }
5588 \msg_new:nnnn { enumext } { starred-unknown-key-value }
5589 {
5590   The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5591   '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5592 }
5593 {
5594   The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5595   ~ have ~ a ~ key ~ called ~ '#1'.\\
5596   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5597 }

```

Message used by unknown $\langle keys \rangle$ in `enumext` environment.

```

5598 \msg_new:nnnn { enumext } { standar-unknown-key }
5599 {
5600   The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_tl
5601   ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5602 }

```

```

5603 {
5604     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5605     ~ have ~ a ~ key ~ called ~ '#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5606     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5607 }
5608 \msg_new:nnnn { enumext } { standar-unknown-key-value }
5609 {
5610     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_
5611     ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5612 }
5613 {
5614     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5615     ~ have ~ a ~ key ~ called ~ '#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5616     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5617 }

```

Message used by unknown *(keys)* in `\foreachkeyans`.

```

5618 \msg_new:nnnn { enumext } { for-key-unknown }
5619 { The~key~'#1'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored.}
5620 {
5621     The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5622     Check~that~you~have~spelled~the~key~name~correctly.
5623 }
5624 \msg_new:nnnn { enumext } { for-key-value-unknown }
5625 { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored. }
5626 {
5627     The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5628     Check~that~you~have~spelled~the~key~name~correctly.
5629 }

```

Messages used by `\getkeyans` command.

```

5630 \msg_new:nnn { enumext } { undefined-storage-anskey }
5631 {
5632     Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
5633 }

```

Messages used by `\miniright` command.

```

5634 \msg_new:nnn { enumext } { missing-miniright }
5635 {
5636     Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\\
5637     The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
5638 }
5639 \msg_new:nnn { enumext } { wrong-miniright-place }
5640 {
5641     Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
5642     Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
5643 }
5644 \msg_new:nnn { enumext } { wrong-miniright-use }
5645 {
5646     Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
5647     '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
5648 }
5649 \msg_new:nnn { enumext } { wrong-miniright-starred }
5650 {
5651     Can't ~ use ~ \c_backslash_str miniright ~ in ~ starred ~ environments ~ \msg_line_context:.
5652 }
5653 \msg_new:nnn { enumext } { many-miniright-used }
5654 {
5655     Can't ~ use ~ \c_backslash_str miniright ~ more ~ than ~ once ~ \msg_line_context:.
5656 }

```

Messages used by `\setenumextmeta` command.

```

5657 \msg_new:nnn { enumext } { unknown-set }
5658 {
5659     Argument ~ [#1] ~ is ~ unknown ~ by ~ \c_backslash_str setenumextmeta ~ \msg_line_context:.
5660 }
5661 \msg_new:nnn { enumext } { already-defined }
5662 {
5663     The ~ key ~ '#1' ~ is ~ already ~ defined ~ \msg_line_context:.
5664 }
5665 \msg_new:nnn { enumext } { prohibited-unknown }
5666 {
5667     The ~ name ~ 'unknown' ~ can't ~ be ~ chosen~ for ~ a ~ meta ~ key ~ \msg_line_context:.

```

5668 }

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

```
5669 \msg_new:nnn { enumext } { nested }
5670 {
5671   The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ \msg_line_con
5672 }
5673 \msg_new:nnn { enumext } { nested-horizontal }
5674 {
5675   The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ in ~ '#1' ~ \
5676 }
5677 \msg_new:nnn { enumext } { item-joined }
5678 {
5679   Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~\msg_line_context:.
5680 }
5681 \msg_new:nnn { enumext } { item-joined-columns }
5682 {
5683   Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.
5684 }
```

12.51 Finish package

Finish package implementation.

```
5685 \file_input_stop:
5686 </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>	226
<code>\+</code>	218
<code>\-</code>	218
<code>\\</code>	234, 2766, 4114, 5303, 5312, 5317, 5337, 5339, 5346, 5348, 5361, 5366, 5371, 5386, 5425, 5427, 5429, 5434, 5435, 5440, 5441, 5459, 5476, 5493, 5498, 5507, 5516, 5522, 5528, 5537, 5542, 5551, 5565, 5575, 5585, 5595, 5605, 5615, 5621, 5627, 5636, 5641, 5646
A	
above	<u>1581</u>
above*	<u>1581</u>
<code>\addvspace</code>	1150, 1179, 1222, 1225, 1393, 1396, 1493, 1499, 1534, 1540, 1561, 1567, 3628, 3768, 3786, 3986, 3989, 4319, 4334, 4380, 4394
after	<u>988</u>
align	<u>534</u>
<code>\Alph</code>	36, <u>41</u>
<code>\Alph</code>	486, 604, 649, 717, 5012
<code>\alph</code>	36, <u>41</u>
<code>\alph</code>	487, 602, 5004
<code>\anskey</code>	12, 74, 76, <u>2584</u>
anskey*	13, <u>2694</u>
<code>\anspic</code>	15, 102, 105, <u>4022</u>
<code>\anspic*</code>	68
<code>\arabic</code>	30, 36
<code>\arabic</code>	485, 601, 648, 4996, 5000, 5016
B	
base-fix	<u>847</u>
<code>\baselineskip</code>	<u>50</u>
<code>\baselineskip</code>	864, 875
before	<u>988</u>
before*	<u>988</u>
below	<u>1581</u>
below*	<u>1581</u>
bool commands:	
<code>\bool_gset_false:N</code>	357, 358, 359, 2870, 2872, 4336, 4340, 4396
<code>\bool_gset_true:N</code>	265, 275, 1091, 2074, 2080, 4305, 4337, 4369, 4397
<code>\bool_if:NTF</code>	425, 437, 454, 1515, 1603, 1617, 1630, 1641, 1652, 1663, 1674, 1685, 1734, 1751, 1756, 1764, 1791, 1829, 1834, 1841, 1845, 1867, 1872, 1880, 1887, 1918, 1926, 2019, 2217, 2227, 2306, 2330, 2337, 2361, 2459, 2481, 2521, 2534, 2538, 2588, 2607, 2631, 2685, 2696, 2785, 2822, 2886, 2919, 2934, 3009, 3020, 3024, 3043, 3056, 3098, 3132, 3175, 3194, 3336, 3351, 3413, 3423, 3455, 3460, 3561, 3609, 3637, 3696, 3751, 3776, 3922, 3984, 4024, 4043, 4089, 4298, 4314, 4320, 4363, 4377, 4381, 4505, 4515, 4598, 4605, 4618, 4645, 4668, 4727, 4737, 4844, 4848, 4874, 4881, 4900, 4912, 4931
<code>\bool_if:nTF</code>	1541, 1568, 3154, 3310, 3371, 3901, 4065, 5036, 5178
<code>\bool_if_p:N</code>	284, 299, 860, 861, 871, 872, 1898, 1899, 1907, 1908, 2032, 2058, 2071, 2072, 2077, 2078, 2394, 2404, 2416, 2431, 2432, 2466, 2507, 2508, 2808, 2996, 2997, 3034, 3035, 3534, 3536, 3547, 4072, 4073
<code>\bool_lazy_all:nTF</code>	282, 297, 2030, 2056, 2392, 2401, 2414, 2429, 3532, 3545
<code>\bool_lazy_and:nnTF</code>	261, 271, 859, 870, 1508, 1897, 1906, 2070, 2076, 2465, 2472, 2506, 2649, 2661, 2807, 2813, 2995
<code>\bool_lazy_or:nnTF</code>	1959, 1966, 3033, 4071, 5201
<code>\bool_new:N</code>	34, 35, 36, 37, 38, 39, 40, 41, 64, 73, 96, 101, 102, 107, 108, 111, 136, 137, 144, 151, 152, 157, 159, 160, 174, 186, 188
<code>\bool_not_p:n</code>	262, 272, 2403, 2467, 2473, 2809, 2814, 3535, 3548
<code>\bool_set_eq:NN</code>	3107, 3290, 4552, 4797
<code>\bool_set_false:N</code>	434, 881, 2004, 2005, 2037, 2042, 2046, 2050, 2063, 2749, 3509, 3654, 3704, 3791, 3919, 3991, 4473, 4500, 4549, 4743, 4794
<code>\bool_set_true:N</code>	289, 290, 304, 305, 416, 420, 527, 896, 1587, 1592, 1854, 1976, 1977, 2249, 2257, 2750, 3101, 3103, 3135, 3137, 3286, 3298, 3435, 3508, 3541, 3554, 3580, 3701, 3728, 3903, 4287, 4352, 4472, 4556, 4563, 4564, 4601, 4741, 4801, 4808, 4809
box commands:	
<code>\box_dp:N</code>	1439, 1440, 1443, 1450, 1463, 1471, 1477, 1485, 3932, 3937, 3986, 4100
<code>\box_ht:N</code>	1222, 1225, 1236, 1237, 1248, 1250, 1265, 1268, 1276, 1277, 1288, 1290, 1305, 1308, 1315, 1316, 1327, 1329, 1344, 1347, 1393, 1396, 1404, 1405, 1413, 1414, 1426, 1428
<code>\box_ht_plus_dp:N</code>	3928, 4052
<code>\box_new:N</code>	70, 147, 148, 181, 187
<code>\box_use_drop:N</code>	4331, 4392, 4665, 4928
<code>\box_wd:N</code>	493
C	
<code>\c</code>	226, 227, 754, 756, 768, 770
<code>\catcode</code>	2766
<code>\cB</code>	227
<code>\cE</code>	227
<code>\centering</code>	1543, 1570, 4012, 4324, 4385
check-ans	<u>1996</u>
Document class:	
article	43
clist commands:	
<code>\clist_const:Nn</code>	193
<code>\clist_map_function:nN</code>	3995
<code>\clist_map_inline:Nn</code>	533, 802, 987, 1002, 1083, 1597
<code>\clist_map_inline:nn</code>	49, 60, 78, 86, 98, 110, 139, 168, 192, 564, 584, 856, 901, 922, 1097, 1703, 1943, 2010, 2196, 2214, 2246, 2389, 2928, 3215, 3227, 3267, 3400, 3403, 3430, 3442, 3445, 3465, 5154
<code>\columnbreak</code>	75
<code>\columnbreak</code>	2469
columns	<u>1067</u>
columns-sep	<u>1067</u>
<code>\columnsep</code>	97
<code>\columnsep</code>	3604, 3749
<code>\columnseprule</code>	97
<code>\columnseprule</code>	3607, 3750
Commands provide by enumext:	
<code>\anskey</code>	28, 65, 70–74, 76, 77, 83, 85, 96, 114, 124, 125, 132

<code>\anspic*</code>	28, 29, 68, 71, 83, 84, 104, 105, 124, 125
<code>\anspic</code>	28, 72, 102, 104, 105, 132
<code>\foreachkeyans</code>	128, 135
<code>\getkeyans</code>	71, 124, 135
<code>\item*</code> 28, 29, 68, 71, 72, 83, 84, 87, 90, 115, 116, 121, 124, 125	
<code>\item</code>	87, 90, 108, 114, 115, 117, 120
<code>\miniright</code>	27, 47, 55, 56, 97, 98, 135
<code>\printkeyans*</code>	124
<code>\printkeyans</code>	28, 72, 124, 125
<code>\setenumextmeta</code>	128, 135
<code>\setenumext</code>	28, 125–127, 131
Counters defined by <code>enumext</code> :	
<code>enumXiii</code>	26, 36
<code>enumXii</code>	26, 36
<code>enumXiv</code>	26, 36
<code>enumXi</code>	26, 36
<code>enumXviii</code>	26, 36
<code>enumXvii</code>	26, 36, 116
<code>enumXvi</code>	26, 36
<code>enumXv</code>	26, 36
cs commands:	
<code>\cs_generate_variant:Nn</code>	198, 199, 495, 511, 760, 776, 2298, 2303, 2379, 2702, 3390, 3997, 5213
<code>\cs_if_exist:NTF</code>	465
<code>\cs_if_free:NTF</code>	2653, 2665
<code>\cs_new:Nn</code>	212
<code>\cs_new:Npn</code>	230, 1704, 1713, 1721, 2261, 2270, 2278, 5062, 5071, 5080
<code>\cs_new_eq:NN</code>	384, 385, 390, 391, 439, 440, 443, 444
<code>\cs_new_protected:Nn</code>	222, 254, 280, 313, 343, 349, 355, 361, 367, 375, 393, 411, 625, 688, 740, 857, 1003, 1007, 1011, 1015, 1019, 1023, 1027, 1031, 1035, 1039, 1043, 1047, 1051, 1055, 1059, 1063, 1098, 1110, 1134, 1152, 1163, 1181, 1207, 1228, 1353, 1379, 1399, 1432, 1454, 1489, 1495, 1598, 1612, 1626, 1637, 1648, 1659, 1670, 1681, 1762, 1865, 1878, 1895, 1916, 1944, 1949, 1974, 2015, 2025, 2068, 2083, 2090, 2099, 2104, 2109, 2114, 2123, 2128, 2133, 2304, 2328, 2335, 2359, 2366, 2380, 2605, 2624, 2640, 2703, 2739, 2770, 2805, 2847, 2868, 2876, 2917, 2932, 2960, 2993, 3029, 3041, 3054, 3140, 3150, 3161, 3169, 3185, 3306, 3322, 3330, 3344, 3484, 3501, 3530, 3559, 3566, 3588, 3618, 3635, 3677, 3694, 3718, 3735, 3760, 3774, 3917, 3993, 3998, 4109, 4117, 4148, 4277, 4296, 4342, 4361, 4401, 4405, 4424, 4459, 4487, 4494, 4503, 4513, 4534, 4683, 4725, 4756, 4762, 4779, 4836, 4946
<code>\cs_new_protected:Npn</code>	200, 204, 208, 236, 447, 463, 480, 490, 496, 605, 650, 722, 747, 761, 1525, 1554, 1730, 1749, 1819, 1852, 1954, 2138, 2215, 2225, 2247, 2255, 2290, 2299, 2455, 2518, 2532, 2570, 2574, 2694, 2725, 2729, 2760, 2896, 2970, 3014, 3094, 3113, 3228, 3232, 3246, 3250, 3268, 3272, 3282, 3294, 3359, 3393, 3433, 3512, 3714, 3893, 3910, 4041, 4061, 4085, 4179, 4228, 4476, 4540, 4547, 4561, 4569, 4574, 4584, 4749, 4785, 4792, 4806, 4814, 4831, 4968, 4981, 5029, 5151, 5163, 5187, 5199, 5237, 5247, 5255, 5277
<code>\cs_new_protected_nopar:Nn</code>	3827, 3869, 3877, 3885, 4523, 4527, 4659, 4768, 4772, 4922
<code>\cs_new_protected_nopar:Npn</code>	3819, 3835, 4590, 4634, 4864, 4889
<code>\cs_set:Npn</code>	2390, 2427, 4974
<code>\cs_set_eq:NN</code>	4449, 4450, 4636, 4713, 4714, 4891
<code>\cs_set_protected:Nn</code>	927, 943, 955, 967
<code>\cs_set_protected:Npn</code>	45, 54, 71, 79, 93, 99, 132, 164, 172, 512, 534, 569, 585, 632, 777, 803, 847, 883, 906, 979, 988, 1067, 1084, 1581, 1692, 1935, 1996, 2155, 2197, 2233, 2382, 2921, 3204, 3220, 3260, 3391, 3431
<code>\cs_to_str:N</code>	482, 505
<code>\cs_undefine:N</code>	2642, 2643, 2644, 2645
D	
<code>\d</code>	218
<code>\DeclareDocumentEnvironment</code>	397
dim commands:	
<code>\dim_abs:n</code>	3364, 3369
<code>\dim_add:Nn</code>	3936, 4142, 4173
<code>\dim_compare:nNnTF</code>	929, 945, 957, 969, 1240, 1252, 1280, 1292, 1319, 1331, 1408, 1416, 1527, 1556, 3361, 3366, 3372, 3378, 3380, 3382, 3571, 3593, 3722, 3739, 3912, 4119, 4135, 4150, 4166, 4279, 4344
<code>\dim_compare:nTF</code>	2491, 2835, 3490, 3683
<code>\dim_eval:n</code>	4096
<code>\dim_gset_eq:NN</code>	4288, 4353
<code>\dim_gzero:N</code>	2874, 4339, 4399
<code>\dim_new:N</code>	67, 74, 75, 76, 95, 141, 149, 150, 180, 182, 183, 189
<code>\dim_set:Nn</code>	493, 897, 3130, 3364, 3369, 3371, 3374, 3375, 3379, 3381, 3384, 3385, 3387, 3486, 3574, 3596, 3679, 3724, 3741, 3926, 4000, 4050, 4121, 4128, 4152, 4159, 4214, 4263, 4281, 4346, 4586
<code>\dim_set_eq:NN</code>	592, 639, 710, 714, 3045, 3046, 3058, 3059, 3125, 3402, 3444, 3604, 3749, 4221, 4224, 4225, 4270, 4273, 4274, 4579, 4655, 4910
<code>\dim_sub:Nn</code>	3495, 3688, 4137, 4168
<code>\dim_use:N</code>	930, 938, 1528, 1538, 2369, 2372, 2377, 3145, 3147, 3190, 3492, 3497, 3572, 3577, 3578, 3584, 3594, 3598, 3599, 3601
<code>\dim_zero:N</code>	3436, 3607, 3750, 3938, 3939, 3940
<code>\dim_zero_new:N</code>	462
<code>\c_zero_dim</code>	932, 946, 958, 970, 1528, 1556, 2493, 2837, 3361, 3366, 3372, 3379, 3492, 3572, 3594, 3685, 3722, 3739, 3912, 4119, 4135, 4150, 4166, 4279, 4344
<code>\dimeval</code>	2162
E	
<code>\end</code>	2332, 2363, 3625, 3765, 3980, 4014, 5038, 5047, 5054
end internal commands:	
<code>\end__enumext_mini_page</code>	1536, 1563, 3646, 3785, 4303, 4367, 4393
<code>\endgroup</code>	2766
<code>\endlist</code>	385
<code>\endminipage</code>	391
<code>enumext</code>	5, 3466
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>	128, 5165, 5181, 5182, 5184, 5187
<code>__enumext_add_pre_parsep:</code>	48, 1108, 1110, 1110
<code>__enumext_after_args_exec:</code>	46, 1003, 1015, 3479
<code>__enumext_after_args_exec_v:</code>	1019, 1031, 3670
<code>__enumext_after_args_exec_viii:</code>	1035, 1059
<code>__enumext_after_args_exec_viii:</code>	1063
<code>__enumext_after_env:nn</code>	80, 81, 83, 98, 110, 118, 204, 204, 2780, 3657, 4312, 4375, 4699
<code>__enumext_after_hyperref:</code>	34, 409, 411, 411
<code>__enumext_after_list:</code>	98, 119, 3482, 3635, 3635
<code>\l__enumext_after_list_args_v_tl</code>	1033

`\l__enumext_after_list_args_vii_tl` 1061, 4654
`\l__enumext_after_list_args_viii_tl` .. 1065, 4909
`__enumext_after_list_v:` 3675, 3718, 3774
`__enumext_after_list_vii:` 114, 4457, 4494, 4494
`__enumext_after_list_viii:` .. 4723, 4762, 4762
`__enumext_after_stop_list:` . 46, 98, 1003, 1011, 3651
`__enumext_after_stop_list_v:` 1019, 1027, 3792
`\l__enumext_after_stop_list_v_tl` 1029
`__enumext_after_stop_list_vii:` .. 114, 1035, 1051, 4497
`\l__enumext_after_stop_list_vii_tl` ... 1053
`__enumext_after_stop_list_viii:` . 1055, 4765
`\l__enumext_after_stop_list_viii_tl` ... 1057
`\l__enumext_align_label_pos_v_str` 3348
`\l__enumext_align_label_pos_X_str` 79
`\l__enumext_align_label_vii_str` .. 4620, 4627
`\l__enumext_align_label_viii_str` 4878
`\l__enumext_align_label_X_str` 172
`\c__enumext_all_envs_clist` .. 193, 533, 802, 987, 1002, 1083, 1597
`\c__enumext_all_families_seq` .. 127, 5119, 5145
`\l__enumext_anskey_env_bool` 31, 79, 34, 290, 305, 2696
`__enumext_anskey_env_clean_vars:` . 82, 2801, 2805, 2868
`__enumext_anskey_env_define_keys:` 79, 2694, 2703, 2774
`__enumext_anskey_env_exec:` 81, 2699, 2770, 2770
`__enumext_anskey_env_make:n` 65, 79, 1979, 2694, 2694, 2702
`__enumext_anskey_env_reset_keys:` 80, 81, 2739, 2802
`__enumext_anskey_env_reset_keys:__enumext_rescan_anskey_env:n` 2694
`__enumext_anskey_env_save_keys:` .. 81, 2782, 2805, 2805
`__enumext_anskey_env_store:` .. 82, 2798, 2805, 2847
`__enumext_anskey_env_unknown:n` 80, 2722, 2725
`__enumext_anskey_env_unknown:nn` . 2727, 2729
`\l__enumext_anskey_level_int` .. 28, 2626, 2627
`__enumext_anskey_safe_inner:` . 78, 2599, 2605, 2624
`__enumext_anskey_safe_inner:n` 77
`__enumext_anskey_safe_outer:` . 77, 2586, 2605, 2605
`__enumext_anskey_show_wrap_arg:n` . 76, 2518, 2518, 2536, 2551
`__enumext_anskey_show_wrap_left:n` 76, 2463, 2532, 2532
`__enumext_anskey_unknown:n` 77, 2554, 2568, 2570
`__enumext_anskey_unknown:nn` . 2554, 2572, 2574
`__enumext_anskey_wrapper:n` 2159, 2530
`\l__enumext_anspic_body_box` ... 140, 4049, 4052
`\l__enumext_anspic_body_htdp_dim` . 140, 4050, 4099
`__enumext_anspic_box_set_dim:n` .. 4041, 4041, 4088
`__enumext_anspic_label:nn` ... 4061, 4091, 4105
`\l__enumext_anspic_label_box` .. 140, 3925, 3928
`\l__enumext_anspic_label_htdp_dim` . 104, 140, 3926, 3932, 4098
`__enumext_anspic_start_list_tag:` 3843, 3869, 4111
`__enumext_anspic_stop_list_tag:` . 3843, 3885, 4115
`__enumext_anspic_stop_start_list_tag:` 3843, 3877, 4113
`__enumext_at_begin_document:n` 33, 34, 200, 200, 382, 388
`\l__enumext_base_line_fix_bool` . 851, 861, 872, 881
`__enumext_before_args_exec:` . 46, 96, 113, 1003, 1003, 3569
`__enumext_before_args_exec_v:` 1019, 1019, 3721
`__enumext_before_args_exec_vii:` . 1035, 1035, 4491
`__enumext_before_args_exec_viii:` 1039, 4759
`__enumext_before_env:nn` 79, 204, 208, 2647, 2659, 2671, 2772
`__enumext_before_keys_exec:` .. 46, 1003, 1007, 3476
`__enumext_before_keys_exec_v:` 1019, 1023, 3667
`__enumext_before_keys_exec_vii` 1035
`__enumext_before_keys_exec_vii:` . 1043, 4443
`__enumext_before_keys_exec_viii:` 1047, 4708
`__enumext_before_list:` ... 96, 3470, 3566, 3566
`__enumext_before_list_v:` ... 3662, 3718, 3718
`__enumext_before_list_vii:` ... 113, 4438, 4487, 4487
`__enumext_before_list_viii:` .. 119, 4704, 4756, 4756
`\l__enumext_before_no_starred_key_v_tl` 1025
`\l__enumext_before_no_starred_key_vii_tl` 1045
`\l__enumext_before_no_starred_key_viii_tl` 1049
`\l__enumext_before_starred_key_v_tl` ... 1021
`\l__enumext_before_starred_key_vii_tl` . 1037
`\l__enumext_before_starred_key_viii_tl` 1041
`__enumext_calc_hspace:NNNNNNN` 92, 3359, 3359, 3390, 3395, 3437
`__enumext_check_ans_active:` . 66, 96, 113, 2015, 2015, 3570, 4490
`\g__enumext_check_ans_item_tl` 85
`\g__enumext_check_ans_key_bool` 67, 68, 151, 357, 2074, 2080, 2886
`\l__enumext_check_ans_key_bool` 67, 2000, 2005, 2071, 2077
`__enumext_check_ans_key_hook:` 67, 98, 114, 2068, 2068, 3652, 4498
`__enumext_check_ans_level:` 66, 2015, 2021, 2025
`__enumext_check_ans_log:` 67, 68, 83, 2114, 2114, 2890
`__enumext_check_ans_log_msg_greater:` 2114, 2120, 2133
`__enumext_check_ans_log_msg_less:` 2114, 2118, 2123
`__enumext_check_ans_log_msg_same_ok:` 2114, 2119, 2128
`__enumext_check_ans_msg_greater:` 2090, 2096, 2109
`__enumext_check_ans_msg_less:` 2090, 2094, 2099
`__enumext_check_ans_msg_same_ok:` 2090, 2095, 2104
`__enumext_check_ans_show:` .. 67, 82, 2090, 2090, 2888

`\l__enumext_check_answers_bool` . 65, 66, 77, 87, 151, 1977, 2004, 2019, 2306, 2330, 2337, 2361, 2588, 2785, 3009, 3098, 3132, 4598
`__enumext_check_starred_cmd:n` 32, 68, 85, 118, 2138, 2138, 3673, 3982, 4721
`\g__enumext_check_starred_cmd_int` 151, 2141, 2147, 2152, 3304, 4070, 4843
`\l__enumext_check_start_line_env_tl` . 32, 151, 320, 328, 336, 2144, 2150, 2153
`\l__enumext_columns_sep_v_dim` 3739, 3741, 3749
`\l__enumext_columns_sep_vii_dim` .. 4119, 4121, 4130, 4142, 4218, 4680
`\l__enumext_columns_sep_viii_dim` . 4150, 4152, 4161, 4173, 4267, 4943
`\l__enumext_columns_v_int` 1373, 1391, 1559, 3737, 3745, 3757, 3762
`\l__enumext_columns_vii_int` .. 4124, 4127, 4131, 4140, 4182, 4186, 4189, 4195, 4201, 4205, 4674, 4688
`\l__enumext_columns_viii_int` . 4155, 4158, 4162, 4171, 4231, 4235, 4238, 4244, 4250, 4254, 4937, 4952
`\l__enumext_counter_i_tl` 45, 472
`\l__enumext_counter_ii_tl` 45, 473
`\l__enumext_counter_iii_tl` 45, 474
`\l__enumext_counter_iv_tl` 45, 475
`\c__enumext_counter_style_tl` 30, 50, 224
`\g__enumext_counter_styles_tl` . 26, 36, 67, 483, 501
`\l__enumext_counter_v_tl` 45, 476, 730
`\l__enumext_counter_vi_tl` 45, 477
`\l__enumext_counter_vii_tl` 45, 478, 660
`\l__enumext_counter_viii_tl` 45, 479, 677
`\l__enumext_current_widest_dim` 26, 67, 507, 593, 640, 711, 715
`__enumext_def_meta_key:nnn` ... 128, 5165, 5193, 5199, 5213
`__enumext_default_item:n` ... 3094, 3094, 3158
`__enumext_define_counters:Nn` 26, 463, 463, 472, 473, 474, 475, 476, 477, 478, 479
`__enumext_endminipage:` . 34, 388, 391, 405, 4333, 4661, 4924
`\g__enumext_envir_name_tl` 32, 34, 291, 306, 365, 1947, 1952, 1962, 2102, 2107, 2112, 2126, 2131, 2136
`\l__enumext_envir_name_tl` . 31, 32, 34, 260, 270, 319, 327, 335, 5561, 5564, 5571, 5574, 5581, 5584, 5591, 5594, 5600, 5604, 5610, 5614, 5671, 5675
`__enumext_execute_after_env:` 33, 64, 67, 68, 78, 82, 2876, 2876, 3657, 4699
`__enumext_fake_item:` 927, 927, 3422
`\l__enumext_fake_item_indent_v_dim` 946, 951
`\l__enumext_fake_item_indent_v_tl` 948, 3287, 3291, 3299
`\l__enumext_fake_item_indent_vii_dim` 958, 963
`\l__enumext_fake_item_indent_vii_tl` 960, 4657
`\l__enumext_fake_item_indent_viii_dim` . 970, 975, 4916
`\l__enumext_fake_item_indent_viii_tl` .. 972, 4914, 4919
`\l__enumext_fake_item_indent_X_tl` 99
`__enumext_fake_item_vii:` 927, 955, 3454
`__enumext_fake_item_viii:` 927, 967, 3459
`__enumext_fake_make_label_vii:n` . 116, 4590, 4590, 4651
`__enumext_fake_make_label_viii:n` 4864, 4864, 4906
`__enumext_filter_first_level:n` .. 126, 5062, 5062, 5096, 5107
`__enumext_filter_first_level_key:n` 126, 5062, 5067, 5071
`__enumext_filter_first_level_pair:nn` . 126, 5062, 5068, 5080
`__enumext_filter_save_key:n` .. 71, 2222, 2230, 2253, 2259, 2261, 2261, 4994, 4998, 5002, 5006, 5010, 5014
`__enumext_filter_save_key_key:n` .. 71, 2261, 2266, 2270
`__enumext_filter_save_key_pair:nn` 71, 2261, 2267, 2278
`__enumext_filter_series:n` 59, 1704, 1704, 1742, 1754, 1759
`__enumext_filter_series_key:n` 59, 1704, 1709, 1713
`__enumext_filter_series_pair:nn` .. 60, 1704, 1710, 1721
`__enumext_first_item_tmp_vii:` 112, 114, 4449, 4523, 4523
`__enumext_first_item_tmp_viii:` 118, 120, 4713, 4768, 4768
`\g__enumext_footnote_arg_seq` . 169, 4407, 4420, 4430
`\g__enumext_footnote_int` . 169, 4414, 4417, 4419, 4421
`\g__enumext_footnote_int_seq` . 169, 4408, 4421, 4426, 4429
`__enumext_footnotes_key_bool` 34
`\l__enumext_footnotes_key_bool` 29, 35, 117, 159, 420, 425, 434, 4645, 4668, 4900, 4931
`__enumext_footnotetext:nn` ... 4401, 4401, 4431
`__enumext_foreach_add_body:n` . 129, 5214, 5274, 5277
`\l__enumext_foreach_after_tl` 5218, 5286
`\l__enumext_foreach_before_tl` 5216, 5281
`\g__enumext_foreach_default_keys_tl` 128, 125, 5236, 5257
`__enumext_foreach_keyans:nn` .. 129, 5214, 5253, 5255
`\l__enumext_foreach_name_prop_tl` . 125, 5259, 5284
`\l__enumext_foreach_print_seq` 125, 5269, 5275, 5279
`\l__enumext_foreach_sep_tl` 5228, 5275
`\l__enumext_foreach_start_int` 5220, 5271
`\l__enumext_foreach_step_int` 5224, 5272
`\l__enumext_foreach_stop_int` . 5222, 5264, 5266, 5273
`__enumext_foreach_wrapper:n` 5226, 5282
`__enumext_getkeyans:nn` .. 124, 4977, 4981, 4981
`__enumext_getkeyans_aux:n` 124, 4965, 4968, 4968
`\l__enumext_hyperref_bool` . 29, 34, 35, 159, 416, 437, 454, 2508, 2997
`__enumext_hypertarget:nn` 35, 411, 439, 443, 459
`__enumext_if_is_int:n` 216
`__enumext_if_is_int:nTF` 216, 749, 763
`__enumext_internal_mini_page:` 34, 95, 113, 393, 393, 3503, 4461
`__enumext_is_not_nested:` 26, 31, 95, 113, 254, 254, 3504, 4462
`__enumext_is_on_first_level:` . 26, 31, 95, 113, 254, 280, 3510, 4474


```

\g__enumext_item_anskey_int 77, 85, 151, 352, 379,
    380, 2087, 2457, 3011
\__enumext_item_answer_diff: .. 67, 68, 82, 2083,
    2083, 2883
\g__enumext_item_answer_diff_int . 67, 68, 151,
    353, 2085, 2092, 2116
\l__enumext_item_column_pos_vii_int 114, 4189,
    4195, 4201, 4205, 4212, 4530, 4674, 4677
\l__enumext_item_column_pos_viii_int .. 120,
    4238, 4244, 4250, 4254, 4261, 4775, 4937, 4940
l__enumext_item_column_pos_X_int ..... 172
\g__enumext_item_count_all_vii_int 114, 4213,
    4531, 4688, 4696
\g__enumext_item_count_all_viii_int 120, 4262,
    4776, 4951, 4960
\g__enumext_item_count_all_X_int ..... 172
\g__enumext_item_number_bool ..... 151
\l__enumext_item_number_bool 66, 157, 2037, 2042,
    2046, 2050, 2063, 2631, 2685, 3101, 3135, 4601
\g__enumext_item_number_int 66, 67, 151, 351, 378,
    380, 2036, 2041, 2045, 2049, 2062, 2087, 3100, 3134,
    4600
\__enumext_item_peek_args_vii: 114, 115, 4532,
    4534, 4534
\__enumext_item_peek_args_viii: .. 120, 4777,
    4779, 4779
\__enumext_item_star_exec: 88, 3113, 3140, 3177,
    3196
\l__enumext_item_starred_vii_bool 4549, 4563,
    4605
\l__enumext_item_starred_viii_bool 4794, 4808,
    4874, 4912
\l__enumext_item_starred_X_bool ..... 172
\__enumext_item_std:w . 33, 87, 91, 382, 386, 3104,
    3110, 3138, 3287, 3291, 3299
\g__enumext_item_symbol_aux_tl . 87, 129, 3118,
    3121, 3146, 3182, 3200
\g__enumext_item_symbol_aux_vii_tl 4571, 4607,
    4610, 4614, 4616
\g__enumext_item_symbol_aux_X_tl ..... 172
\l__enumext_item_symbol_sep_vii_dim .. 4579,
    4586, 4613, 4615
\l__enumext_item_symbol_vii_tl ..... 4610
\l__enumext_item_text_vii_box .... 4637, 4665
\l__enumext_item_text_viii_box ... 4892, 4928
\l__enumext_item_text_X_box ..... 172
\l__enumext_item_width_vii_dim ... 4128, 4137,
    4216, 4224, 4225
\l__enumext_item_width_viii_dim .. 4159, 4168,
    4265, 4273, 4274
\l__enumext_item_width_X_dim ..... 172
\l__enumext_itemindent_X_dim ..... 71
\l__enumext_itemsep_i_skip ... 1234, 1241, 1244,
    1246, 1253, 1257, 1260, 1262, 1402, 1409, 1411, 1412,
    1417, 1421, 1423, 1424
\l__enumext_itemsep_ii_skip .. 1274, 1281, 1284,
    1286, 1293, 1297, 1300, 1302
\l__enumext_itemsep_iii_skip . 1313, 1320, 1323,
    1325, 1332, 1336, 1339, 1341
\l__enumext_itemsep_vii_skip ..... 4694
\l__enumext_itemsep_viii_skip ..... 4958
\l__enumext_joined_item_aux_vii_int .. 4210,
    4211, 4212, 4213, 4219
\l__enumext_joined_item_aux_viii_int . 4259,
    4260, 4261, 4262, 4268
\l__enumext_joined_item_aux_X_int .... 172
\__enumext_joined_item_vii:w .. 115, 4537, 4538,
    4540, 4540
\l__enumext_joined_item_vii_int .. 4181, 4182,
    4185, 4187, 4193, 4198, 4203, 4208, 4210, 4216
\__enumext_joined_item_viii:w . 120, 4782, 4783,
    4785, 4785
\l__enumext_joined_item_viii_int . 4230, 4231,
    4234, 4236, 4242, 4247, 4252, 4257, 4259, 4265
\l__enumext_joined_item_X_int ..... 172
\l__enumext_joined_width_vii_dim . 4214, 4221,
    4224, 4639, 4653
\l__enumext_joined_width_viii_dim 4263, 4270,
    4273, 4894, 4908
\l__enumext_joined_width_X_dim ..... 172
\__enumext_keyans_addto_prop:n 83, 2896, 2896,
    3301, 4067
\__enumext_keyans_addto_seq:n . 84, 2970, 2970,
    3303, 4069
\__enumext_keyans_addto_seq_link: 2970, 2991,
    2993, 4842
\__enumext_keyans_anspic_code:nnn 106, 4038,
    4041, 4109
\__enumext_keyans_anspic_label:nnn 4041, 4085,
    4112
\__enumext_keyans_default_item:n .. 90, 3282,
    3282, 3318
\l__enumext_keyans_env_bool 34, 3535, 3548, 3701,
    3791
\__enumext_keyans_fake_item: .. 927, 943, 3412
\l__enumext_keyans_level_h_int .. 119, 28, 670,
    697, 2615, 2677, 2948, 4468, 4731, 4732
\l__enumext_keyans_level_int .. 28, 1519, 2611,
    2673, 2943, 3700, 3705, 4032
\__enumext_keyans_make_label: 37, 91, 3306, 3322,
    3322, 3410
\__enumext_keyans_make_label_box: 3322, 3326,
    3344
\__enumext_keyans_make_label_std: 3322, 3328,
    3330
\__enumext_keyans_mini_right_cmd:n 56, 1521,
    1554, 1554
\__enumext_keyans_mini_set_vskip: ..... 53
\__enumext_keyans_minipage_add_space: 1353,
    1379, 3730
\__enumext_keyans_minipage_set_skip: . 1353,
    1353, 1381
\__enumext_keyans_multi_addvspace: 1152, 1163,
    3754
\__enumext_keyans_multi_set_vskip: 49, 1152,
    1152, 1165
\__enumext_keyans_multicols_start: 3718, 3733,
    3735
\__enumext_keyans_multicols_stop: 1558, 3718,
    3760, 3789
\__enumext_keyans_name_and_start: 26, 32, 119,
    313, 313, 3702, 3900, 4736
\__enumext_keyans_parse_keys:n 3661, 3714, 3714
\l__enumext_keyans_pic_above_int . 140, 4001,
    4002, 4004
\__enumext_keyans_pic_arg_two: 103, 3917, 3917,
    3947
\l__enumext_keyans_pic_below_int . 140, 4001,

```

4002, 4005
 \l__enumext_keyans_pic_body_seq 104–106, 140, 3973, 4013, 4036
 __enumext_keyans_pic_do:n 105, 3973, 3975, 3993, 3993, 3997
 \l__enumext_keyans_pic_label_pos_str .. 103, 140, 3904, 3907, 4011
 \l__enumext_keyans_pic_level_int .. 28, 1503, 2619, 2681, 2899, 2938, 2973, 3061, 3895, 3896
 \g__enumext_keyans_pic_parsep_skip 140, 3934, 3989
 __enumext_keyans_pic_row:n ... 105, 3995, 3998, 3998
 __enumext_keyans_pic_safe_exec:n 103, 3893, 3893, 3946
 __enumext_keyans_pic_skip_abs:N . 103, 3910, 3910, 3921
 \l__enumext_keyans_pic_star_bool .. 103, 140, 3903, 3922, 3984, 4043, 4089
 \l__enumext_keyans_pic_width_dim . 140, 4000, 4011, 4063
 __enumext_keyans_pre_itemsep_skip: .. 1353, 1372, 1399
 __enumext_keyans_redefine_item: .. 91, 3306, 3306, 3409
 __enumext_keyans_ref: 41, 722, 740, 3411
 __enumext_keyans_ref:n 41, 719, 722, 722
 __enumext_keyans_safe_exec: . 3660, 3694, 3694
 __enumext_keyans_set_item_width: . 99, 3669, 3677, 3677
 __enumext_keyans_show_ans: .. 3014, 3022, 3041
 __enumext_keyans_show_item_opt: . 3014, 3029, 3299, 4082, 4915
 __enumext_keyans_show_left:n . 91, 3014, 3014, 3297, 4076
 __enumext_keyans_show_pos: .. 3014, 3026, 3054
 __enumext_keyans_starred_item:n .. 91, 3294, 3294, 3314
 __enumext_keyans_store_ref: .. 83, 2917, 2917, 3302, 4068, 4840
 __enumext_keyans_store_ref_aux_i: 84, 2917, 2929, 2932
 __enumext_keyans_store_ref_aux_ii: 84, 2917, 2958, 2960
 __enumext_keyans_unknown_keys:n . 3220, 3224, 3228
 __enumext_keyans_unknown_keys:nn 3220, 3230, 3232
 __enumext_keyans_wrapper_opt:n .. 2165, 3037
 \l__enumext_label_copy_i_tl .. 2423, 2936, 2941, 2946, 2951
 \l__enumext_label_copy_v_tl 2946
 \l__enumext_label_copy_vi_tl 2941
 \l__enumext_label_copy_vii_tl 2399, 2410, 2439, 2936
 \l__enumext_label_copy_viii_tl 2951
 \l__enumext_label_copy_X_tl 161
 \l__enumext_label_fill_left_v_tl 3334
 \l__enumext_label_fill_left_X_tl 99
 \l__enumext_label_fill_right_v_tl 3341
 \l__enumext_label_fill_right_X_tl 99
 \l__enumext_label_font_style_v_tl 3335, 3350, 4080
 \l__enumext_label_font_style_vii_tl .. 4622, 4629
 \l__enumext_label_font_style_viii_tl .. 4880
 \l__enumext_label_i_tl 585
 \l__enumext_label_ii_tl 585
 \l__enumext_label_iii_tl 585
 \l__enumext_label_iv_tl 585
 __enumext_label_style:Nnn 26, 36, 496, 496, 511, 590, 637, 708, 712
 \l__enumext_label_v_tl .. 83, 84, 705, 2904, 2978, 3048, 3088, 3296, 3300, 3664, 3925, 4075, 4077
 \l__enumext_label_vi_tl . 83, 84, 705, 2901, 2975, 4075, 4077, 4081
 \l__enumext_label_vii_tl . 632, 4558, 4581, 4588
 \l__enumext_label_viii_tl 632, 4803, 4834, 4838
 \l__enumext_label_width_by_box .. 67, 492, 493
 __enumext_label_width_by_box:Nn 36, 490, 490, 495, 507, 773
 \l__enumext_labelsep_i_dim ... 3046, 3051, 3059, 3091, 4846, 4861
 \l__enumext_labelsep_v_dim 3744
 \l__enumext_labelsep_vii_dim . 2523, 3046, 3059, 4123, 4133, 4217, 4525, 4579, 4632, 4641
 \l__enumext_labelsep_viii_dim 4154, 4164, 4266, 4770, 4887, 4896, 4916
 \l__enumext_labelwidth_i_dim . 3045, 3051, 3058, 3091, 4846, 4861
 \l__enumext_labelwidth_v_dim 3348, 3744
 \l__enumext_labelwidth_vii_dim ... 2523, 3045, 3058, 4123, 4132, 4217, 4525, 4620, 4627, 4640
 \l__enumext_labelwidth_viii_dim .. 4154, 4163, 4266, 4770, 4878, 4895
 \l__enumext_leftmargin_tmp_v_bool . 103, 3919
 \l__enumext_leftmargin_tmp_X_bool 71
 \l__enumext_leftmargin_tmp_X_dim 71
 \l__enumext_leftmargin_X_dim 71
 __enumext_level: 212, 212, 614, 617, 618, 627, 629, 930, 934, 938, 1005, 1009, 1013, 1017, 1100, 1102, 1104, 1106, 1139, 1141, 1143, 1145, 1150, 1185, 1191, 1196, 1198, 1201, 1204, 1217, 1220, 1528, 1532, 1538, 1601, 1603, 1605, 1608, 1615, 1617, 1619, 1622, 2217, 2219, 2221, 2249, 2250, 2252, 2308, 2316, 2320, 2324, 2527, 2528, 3103, 3104, 3108, 3109, 3110, 3118, 3126, 3127, 3130, 3137, 3138, 3142, 3145, 3147, 3173, 3174, 3175, 3178, 3181, 3190, 3191, 3193, 3194, 3197, 3473, 3475, 3492, 3497, 3541, 3554, 3561, 3572, 3574, 3577, 3578, 3580, 3584, 3591, 3594, 3596, 3598, 3599, 3600, 3601, 3604, 3609, 3615, 3621, 3628, 3637
 \l__enumext_level_h_int 113, 28, 263, 286, 300, 653, 690, 1510, 2033, 2053, 2418, 2651, 2663, 3549, 4463, 4464
 \l__enumext_level_int . 95, 28, 214, 273, 285, 301, 395, 1112, 1230, 1509, 2027, 2059, 2395, 2405, 2411, 2417, 2424, 2433, 2438, 2650, 2662, 2878, 3425, 3505, 3506, 3517, 3525, 3539, 3552, 3605, 3709, 4028, 4507, 4517, 4744, 5601, 5605, 5611, 5615
 __enumext_list_arg_two_i: 3391
 __enumext_list_arg_two_ii: 3391
 __enumext_list_arg_two_iii: 3391
 __enumext_list_arg_two_iv: 3391
 __enumext_list_arg_two_v: . 91, 3391, 3666, 3920
 __enumext_list_arg_two_vii: 3431, 4442
 __enumext_list_arg_two_viii: 3431, 4707
 \l__enumext_listoffset_v_dim . 3685, 3690, 3746
 \l__enumext_listparindent_vii_dim 4655
 \l__enumext_listparindent_viii_dim ... 4910

```

\__enumext_log_answer_vars: . 33, 367, 375, 2885
\__enumext_log_global_vars: . 33, 367, 367, 2884
\__enumext_make_label: . 37, 88, 3161, 3161, 3420
\__enumext_make_label_box: . . . 3161, 3165, 3185
\__enumext_make_label_std: . . . 3161, 3167, 3169
\l__enumext_mark_answer_sym_tl 73, 2171, 2374,
    2540, 3063, 3076, 4850
\l__enumext_mark_position_str 129, 2175, 2176,
    2202, 2203, 2372
\l__enumext_mark_ref_sym_tl . . 2188, 2513, 3005
\l__enumext_meta_path_tl . 125, 5189, 5190, 5192,
    5193
\c__enumext_meta_paths_prop . . . . . 128, 5165
\__enumext_mini_addvspace_vii: 55, 1489, 1489,
    4291
\__enumext_mini_addvspace_viii: 55, 1489, 1495,
    4356
__enumext_mini_env* . . . . . 393
\__enumext_mini_page 1538, 1565, 3584, 3731, 4293,
    4358, 4379
\__enumext_mini_right_cmd:n 56, 1523, 1525, 1525
\__enumext_mini_set_vskip_vii: 54, 1432, 1432,
    1491
\__enumext_mini_set_vskip_viii: 54, 1432, 1454,
    1497
\__enumext_minipage:w 34, 388, 390, 399, 4316, 4653,
    4908
\l__enumext_minipage_active_v_bool 3728, 3751,
    3776
\g__enumext_minipage_active_vii_bool . . 110,
    4305, 4314, 4336
\l__enumext_minipage_active_vii_bool . 4287,
    4298
\g__enumext_minipage_active_viii_bool 4369,
    4377, 4396
\l__enumext_minipage_active_viii_bool 4352,
    4363
\g__enumext_minipage_active_X_bool . . . 172
\l__enumext_minipage_active_X_bool . . . . 87
\__enumext_minipage_add_space: . . 51, 97, 1181,
    1207, 3582
\g__enumext_minipage_after_skip 87, 1436, 1448,
    4334, 4394
\l__enumext_minipage_after_skip . . 50, 98, 87,
    1194, 1234, 1236, 1241, 1244, 1248, 1253, 1257, 1260,
    1264, 1276, 1281, 1284, 1288, 1293, 1297, 1300, 1304,
    1315, 1320, 1323, 1327, 1332, 1336, 1339, 1343, 1355,
    1369, 1402, 1404, 1409, 1411, 1413, 1417, 1421, 1423,
    1425, 1456, 1469, 1483, 1534, 1561, 3786
\g__enumext_minipage_center_vii_bool . 4320,
    4337
\g__enumext_minipage_center_viii_bool 4381,
    4397
\g__enumext_minipage_center_X_bool . . . 172
\l__enumext_minipage_hsep_v_dim . . . . . 3726
\l__enumext_minipage_hsep_vii_dim . . . . 4285
\l__enumext_minipage_hsep_viii_dim . . . 4350
\l__enumext_minipage_left_skip 87, 1356, 1434,
    1439, 1443, 1457, 1461, 1475, 1493, 1499
\l__enumext_minipage_left_v_dim . . 3724, 3731
\l__enumext_minipage_left_vii_dim 4281, 4293
\l__enumext_minipage_left_viii_dim 4346, 4358
\l__enumext_minipage_left_X_dim . . . . . 87
\g__enumext_minipage_right_skip 87, 1435, 1440,
    1444, 4319, 4380
\l__enumext_minipage_right_skip . 50, 87, 1183,
    1189, 1194, 1196, 1198, 1357, 1358, 1364, 1369, 1370,
    1371, 1376, 1458, 1465, 1479, 1540, 1567
\l__enumext_minipage_right_v_dim . 1556, 1565,
    3722, 3726
\g__enumext_minipage_right_vii_dim 110, 4289,
    4316, 4339
\l__enumext_minipage_right_vii_dim 110, 4279,
    4284, 4290
\g__enumext_minipage_right_viii_dim . . 4354,
    4379, 4399
\l__enumext_minipage_right_viii_dim . . 4344,
    4349, 4355
\g__enumext_minipage_right_X_dim . . . . . 172
\g__enumext_minipage_right_X_skip . . . . 172
\__enumext_minipage_set_skip: . 50, 1181, 1181,
    1209
\g__enumext_minipage_stat_int 97, 87, 1545, 1572,
    3581, 3639, 3644, 3729, 3778, 3783
\l__enumext_miniright_code_vii_box 4327, 4331
\g__enumext_miniright_code_vii_tl 110, 4322,
    4329, 4338
\l__enumext_miniright_code_viii_box . . 4388,
    4392
\g__enumext_miniright_code_viii_tl 4383, 4390,
    4398
\l__enumext_miniright_code_X_box . . . . . 172
\__enumext_multi_addvspace: . 49, 97, 1134, 1134,
    3612
\__enumext_multi_set_vskip: 48, 1098, 1098, 1136
\l__enumext_multicols_above_ii_skip . . . 1117
\l__enumext_multicols_above_iii_skip . . 1123
\l__enumext_multicols_above_iv_skip . . . 1129
\l__enumext_multicols_above_v_skip 1154, 1168,
    1179, 1370
\l__enumext_multicols_above_X_skip . . . . 79
\l__enumext_multicols_below_ii_skip . . 1237,
    1246, 1250, 1262, 1267
\l__enumext_multicols_below_iii_skip . 1277,
    1286, 1290, 1302, 1307
\l__enumext_multicols_below_iv_skip . . 1316,
    1325, 1329, 1341, 1346
\l__enumext_multicols_below_v_skip 1158, 1172,
    1371, 1405, 1412, 1414, 1424, 1427, 3768
\l__enumext_multicols_below_X_skip . . . . 79
\g__enumext_multicols_right_X_skip . . . . 79
\__enumext_multicols_start: 97, 3586, 3588, 3588
\__enumext_multicols_stop: 97, 1530, 3618, 3618,
    3649
\__enumext_nested_base_line_fix: . 43, 95, 113,
    847, 857, 3521, 4484
\__enumext_newlabel:nn 29, 35, 74, 447, 447, 2449,
    2964
\l__enumext_newlabel_arg_one_tl 29, 35, 74, 84,
    161, 2442, 2450, 2512, 2953, 2965, 3003
\l__enumext_newlabel_arg_two_tl 29, 35, 73, 161,
    2398, 2408, 2421, 2436, 2451, 2940, 2945, 2950, 2966
\__enumext_parse_foreach_keys:n . . 5214, 5230,
    5247
\__enumext_parse_foreach_keys:nn . 5214, 5237,
    5249
\__enumext_parse_keys:n 43, 60, 3469, 3512, 3512
\__enumext_parse_keys_vii:n . 43, 60, 4437, 4476,

```

4476

__enumext_parse_keys_viii:n . 4703, 4749, 4749

__enumext_parse_save_key:n 70, 2242, 2247, 2247

__enumext_parse_save_key_vii:n 70, 2237, 2247, 2255

__enumext_parse_series:n 60, 95, 113, 1730, 1730, 3520, 4482

__enumext_parse_store_keys:n 95

\l__enumext_parsep_i_skip 1115, 1117

\l__enumext_parsep_ii_skip 1121, 1123

\l__enumext_parsep_iii_skip 1127, 1129

\l__enumext_parsep_vii_skip 4656

\l__enumext_parsep_viii_skip 4911

\l__enumext_partopsep_v_skip . 1170, 1174, 1366, 1389

\l__enumext_partopsep_viii_skip 1467

__enumext_phantomsection: 35, 411, 440, 444, 460

__enumext_pre_itemsep_skip: 50, 51, 1199, 1228, 1228

__enumext_print_footnote: . . . 4401, 4424, 4670, 4933

__enumext_print_keyans_box:NN 73, 2366, 2366, 2379, 2523, 2526, 3050, 3090, 4846, 4861

\l__enumext_print_keyans_i_tl 4999, 5021

\l__enumext_print_keyans_ii_tl . . . 5003, 5022

\l__enumext_print_keyans_iii_tl . . 5007, 5023

\l__enumext_print_keyans_iv_tl . . . 5011, 5024

\l__enumext_print_keyans_starred_tl 124, 125, 129, 4995, 5043

\l__enumext_print_keyans_vii_tl 124, 5015, 5025

\l__enumext_print_keyans_X_tl 129

__enumext_printkeyans:nnn 125, 5026, 5029, 5029

__enumext_redefine_item: . 88, 3150, 3150, 3419

\l__enumext_ref_key_arg_tl . 38, 39, 50, 227, 607, 608, 621, 652, 655, 666, 672, 683, 724, 725, 736

\l__enumext_ref_the_count_tl . 39, 50, 614, 617, 620, 660, 662, 665, 677, 679, 682, 730, 732, 735

__enumext_regex_counter_style: . . 30, 38, 222, 222, 615, 661, 678, 731

__enumext_register_counter_style:Nn . . 480, 480, 485, 486, 487, 488, 489

__enumext_remove_extra_parsep_vii: . . 4456, 4683, 4683

__enumext_remove_extra_parsep_viii: . 4720, 4946, 4946

__enumext_renew_footnote: . . . 4401, 4405, 4647, 4902

\l__enumext_renew_the_count_v_tl 733, 742, 744

\l__enumext_renew_the_count_vii_tl 663, 692, 694

\l__enumext_renew_the_count_viii_tl 680, 699, 701

\l__enumext_renew_the_count_X_tl 50

__enumext_rescan_anskey_env:n . . 80, 82, 2760, 2855, 2863

__enumext_reset_global_bool: . . 343, 346, 355

__enumext_reset_global_int: . . . 343, 345, 349

__enumext_reset_global_tl: 343, 347, 361

__enumext_reset_global_vars: . 33, 83, 343, 343, 2893

\l__enumext_resume_active_bool 60, 62, 61, 1734, 1854

__enumext_resume_counter: . 62, 1852, 1858, 1865

__enumext_resume_counter:n . 60, 62, 1823, 1828, 1852, 1852, 1922, 1930

__enumext_resume_counter_save_ans: . . 62, 63, 1852, 1863, 1895

__enumext_resume_counter_series: 62, 63, 1852, 1861, 1878

\g__enumext_resume_int . . . 61, 1775, 1869, 1870

__enumext_resume_last:n . . 60, 1730, 1736, 1749

\l__enumext_resume_name_tl 61, 1771, 1779, 1782, 1798, 1806, 1809, 1855, 1856, 1884, 1891

__enumext_resume_save_counter: . . 61, 98, 114, 1762, 1762, 3655, 4501

__enumext_resume_series:n . 62, 1698, 1819, 1819

__enumext_resume_starred: . 63, 1699, 1916, 1916

\g__enumext_resume_vii_int 61, 1802, 1874, 1875

\l__enumext_rightmargin_vii_dim . . 4135, 4139, 4144

\l__enumext_rightmargin_viii_dim . 4166, 4170, 4175

__enumext_safe_exec: . . 34, 95, 3468, 3501, 3501

__enumext_safe_exec_vii: . 34, 4436, 4459, 4459

__enumext_safe_exec_viii: 119, 4702, 4725, 4725

\l__enumext_series_name_tl 62

\l__enumext_series_str . . 61, 95, 113, 1696, 1732, 1740, 1741, 1743, 1745, 1766, 1769, 1773, 1793, 1796, 1800, 3516, 4480

__enumext_set_error:nn 5151, 5161, 5163

__enumext_set_item_width: . 95, 3478, 3484, 3484

__enumext_set_parse:n 5135, 5151, 5151

\l__enumext_setkey_tmpa_int . . . 120, 5128, 5132

\l__enumext_setkey_tmpa_seq . . 120, 5126, 5136, 5142, 5144, 5146, 5158

\l__enumext_setkey_tmpa_tl 120, 5134, 5138

\l__enumext_setkey_tmpb_seq . . 120, 5127, 5130, 5134, 5135

\l__enumext_setkey_tmpb_tl 120, 5153, 5155, 5156

\l__enumext_show_answer_bool . 2182, 2206, 2534, 3020, 3034, 4072, 4844

__enumext_show_length:nnn . . 45, 230, 230, 5372, 5373, 5374, 5375, 5376, 5377, 5378, 5379, 5380, 5381, 5387, 5388, 5389, 5390, 5391, 5392, 5393, 5394, 5395, 5396

\l__enumext_show_position_bool . . . 2185, 2209, 2538, 3024, 3035, 4073, 4848

\g__enumext_standar_bool 31, 95, 34, 262, 265, 284, 358, 1764, 1829, 1841, 1867, 1880, 1918, 2058, 2072, 2403, 2416, 2431, 3536

\l__enumext_standar_bool . 95, 98, 34, 2404, 3508, 3654, 4473

\l__enumext_standar_first_bool 31, 95, 34, 289, 860, 1751, 1898, 1960, 1967

__enumext_standar_item_vii:w . 115, 4545, 4547, 4547

__enumext_standar_item_viii:w 120, 4790, 4792, 4792

__enumext_standar_ref: 39, 605, 625, 3421

__enumext_standar_ref:n 38, 597, 605, 605

\g__enumext_standar_series_tl . 61, 1753, 1754, 1920, 1923

__enumext_standar_unknown_keys:n 3260, 3264, 3268

__enumext_standar_unknown_keys:nn 3260, 3270, 3272

\g__enumext_starred_bool 31, 113, 34, 272, 275, 299, 359, 1791, 1834, 1845, 1872, 1887, 1926, 2032, 2078,

2394, 2934, 4340
 \l__enumext_starred_bool 113, 114, 119, 34, 1515,
 2432, 2467, 2473, 2521, 2809, 2814, 3043, 3056, 3509,
 4472, 4500, 4737, 4741
 __enumext_starred_columns_set_vii: . 4117,
 4117, 4447
 __enumext_starred_columns_set_viii: . 4117,
 4148, 4711
 \l__enumext_starred_first_bool 31, 113, 34, 304,
 871, 1756, 1907, 1960, 1967
 __enumext_starred_item:nn . . 3113, 3113, 3156
 __enumext_starred_item_exec: . 121, 4836, 4836,
 4876
 __enumext_starred_item_vii:w . 115, 4544, 4561,
 4561
 __enumext_starred_item_vii_aux_i:w . 4561,
 4566, 4569
 __enumext_starred_item_vii_aux_ii:w . 4561,
 4567, 4572, 4574
 __enumext_starred_item_vii_aux_iii:w 4561,
 4577, 4584
 __enumext_starred_item_viii:w 120, 121, 4789,
 4806, 4806
 __enumext_starred_item_viii_aux_i:w . 121,
 4806, 4811, 4814
 __enumext_starred_item_viii_aux_ii:w . 121,
 4806, 4812, 4829, 4831
 __enumext_starred_joined_item_vii:n 108, 115,
 4179, 4179, 4542
 __enumext_starred_joined_item_viii:n . 108,
 120, 4179, 4228, 4787
 __enumext_starred_ref: 40, 650, 688, 3451
 __enumext_starred_ref:n 39, 644, 650, 650
 \g__enumext_starred_series_tl . 61, 1758, 1759,
 1928, 1931
 __enumext_starred_unknown_keys:n 3242, 3244,
 3246
 __enumext_starred_unknown_keys:nn 3242, 3248,
 3250
 __enumext_start_from:NNn 41, 747, 747, 760, 782,
 788
 \l__enumext_start_i_int 1870, 1882, 1901
 __enumext_start_item_tmp_vii: 112, 4450, 4527,
 4527
 __enumext_start_item_tmp_viii: . . 118, 4714,
 4772, 4772
 __enumext_start_item_vii:w 115, 117, 4553, 4558,
 4581, 4588, 4634, 4634
 __enumext_start_item_viii:w . 120, 4798, 4803,
 4834, 4889, 4889
 \g__enumext_start_line_tl 31, 34, 292, 307, 364,
 2102, 2107, 2112, 2126, 2131, 2136
 __enumext_start_list:nn . 33, 92, 382, 384, 3472,
 3663, 4440, 4705
 __enumext_start_list_tag:n . 3795, 3819, 4650,
 4905
 __enumext_start_mini_vii: 113, 4277, 4277, 4492
 __enumext_start_mini_viii: . . 119, 4342, 4342,
 4760
 __enumext_start_save_ans_msg: 64, 1944, 1944,
 1969
 __enumext_start_store_level: . 96, 3471, 3530,
 3530
 __enumext_start_store_level_vii: 114, 4439,
 4503, 4503
 \l__enumext_start_vii_int . . 1875, 1889, 1910
 \l__enumext_start_X_int 99
 __enumext_stop_item_tmp_vii: . . 112, 114, 117,
 4449, 4455, 4529, 4636
 __enumext_stop_item_tmp_viii: 118, 120, 4713,
 4719, 4774, 4891
 __enumext_stop_item_vii: 117, 4636, 4659, 4659
 __enumext_stop_item_viii: 123, 4891, 4922, 4922
 __enumext_stop_list: . . 33, 382, 385, 3623, 3631,
 3674, 3764, 3771, 4300, 4308, 4365, 4372, 4722
 __enumext_stop_list_tag:n . . 3795, 3835, 4662,
 4925
 __enumext_stop_mini_vii: 110, 114, 4277, 4296,
 4496
 __enumext_stop_mini_viii: 119, 4342, 4361, 4764
 __enumext_stop_save_ans_msg: . 64, 1944, 1949,
 2882
 __enumext_stop_start_list_tag: . . 3795, 3827,
 4652, 4907
 __enumext_stop_store_level: . . 96, 3530, 3559,
 3624, 3632
 __enumext_stop_store_level_vii: . 114, 4301,
 4309, 4503, 4513
 \l__enumext_store_active_bool 28, 65, 111, 1899,
 1908, 1976, 2607, 3534, 3547, 3696, 3704, 3991, 4024,
 4505, 4515, 4727, 4743
 __enumext_store_active_keys:n . . 70, 95, 2215,
 2215, 3527
 __enumext_store_active_keys_vii:n . 70, 113,
 2215, 2225, 4483
 __enumext_store_addto_prop:n 71, 83, 2290, 2290,
 2298, 2458, 2915, 4839
 __enumext_store_addto_seq:n 72, 85, 2299, 2299,
 2303, 2310, 2324, 2332, 2341, 2355, 2363, 2516, 3008
 \l__enumext_store_anskey_arg_tl . . 28, 75, 111,
 2464, 2469, 2471, 2476, 2483, 2486, 2496, 2501, 2504,
 2510, 2516
 __enumext_store_anskey_code:n 74, 77, 82, 2455,
 2455, 2600, 2853, 2861
 \l__enumext_store_anskey_env_tl . . 28, 81, 111,
 2783, 2787, 2793, 2855, 2863
 \l__enumext_store_anskey_opt_tl 28, 81, 82, 111,
 2784, 2811, 2817, 2824, 2830, 2840, 2850, 2859
 __enumext_store_anskey_safe_outer: 77
 \g__enumext_store_columns_break_bool . 2707,
 2808, 2870
 \l__enumext_store_columns_break_bool . 2466,
 2556
 \l__enumext_store_current_label_tl 28, 83–85,
 121, 111, 2898, 2901, 2904, 2911, 2913, 2915, 2972,
 2975, 2978, 2984, 2989, 2999, 3008, 4816, 4821, 4825,
 4838, 4839, 4841
 \l__enumext_store_current_label_tmp_tl . 28,
 111, 3296, 3300
 \l__enumext_store_current_opt_arg_tl 28, 121,
 111, 3018, 3031, 3037, 4827
 __enumext_store_internal_ref: . . 73, 74, 2380,
 2380, 2461
 \g__enumext_store_item_join_int . . 2710, 2815,
 2819, 2871
 \l__enumext_store_item_join_int . . 2474, 2478,
 2559
 \g__enumext_store_item_star_bool . 2712, 2822,
 2872

```

\l__enumext_store_item_star_bool . 2481, 2561
\g__enumext_store_item_symbol_sep_dim 2717,
    2837, 2842, 2874
\l__enumext_store_item_symbol_sep_dim 2493,
    2498, 2566
\g__enumext_store_item_symbol_tl . 2715, 2828,
    2832, 2873
\l__enumext_store_item_symbol_tl . 2484, 2488,
    2564
\l__enumext_store_keyans_item_opt_sep_
    tl . . . . 2168, 2909, 2911, 2982, 2986, 4819, 4823
\__enumext_store_level_close: . 72, 2304, 2328,
    3563
\__enumext_store_level_close_vii: . 72, 2335,
    2359, 4519
\__enumext_store_level_open: 72, 96, 2304, 2304,
    3542, 3555
\__enumext_store_level_open_vii: . 72, 2335,
    2335, 4509
\g__enumext_store_name_tl 28, 65, 111, 363, 370,
    371, 372, 373, 1952, 1978, 2101, 2106, 2111, 2125,
    2130, 2135, 2880
\l__enumext_store_name_tl 28, 64, 66, 111, 1785,
    1788, 1812, 1815, 1903, 1912, 1947, 1956, 1957, 1978,
    1979, 1980, 1982, 1983, 1985, 1987, 1988, 1990, 1992,
    1993, 2017, 2292, 2294, 2301, 2444, 2445, 2546, 2789,
    2955, 2956, 3069, 3082, 4856
\l__enumext_store_ref_key_bool 74, 2191, 2459,
    2507, 2919, 2996
\l__enumext_store_save_key_vii_bool . 2227,
    2257
\l__enumext_store_save_key_vii_tl 2229, 2230,
    2258, 2259, 2339, 2347, 2351, 2355
\l__enumext_store_save_key_X_bool . 70, 129
\l__enumext_store_save_key_X_tl . . . . 70, 129
\l__enumext_store_upper_level_X_bool . 129
\__enumext_storing_exec: . 64, 65, 79, 1954, 1970,
    1974
\__enumext_storing_set:n . 64, 1939, 1954, 1954
\l__enumext_the_counter_v_tl . . . . . 732
\l__enumext_the_counter_vii_tl . . . . . 662
\l__enumext_the_counter_viii_tl . . . . . 679
\l__enumext_the_counter_X_tl . . . . . 50
\__enumext_tmp:n 45, 49, 54, 60, 71, 78, 79, 86, 93, 98,
    99, 110, 132, 139, 164, 168, 172, 192, 847, 856, 1692,
    1703, 1935, 1943, 1996, 2014, 2155, 2196, 2197, 2214,
    2233, 2246, 2382, 2389, 2390, 2411, 2424, 2427, 2438,
    2921, 2928, 3220, 3227, 3260, 3267, 3391, 3430, 3431,
    3465
\__enumext_tmp:nn 512, 533, 534, 568, 569, 584, 777,
    802, 883, 905, 906, 926, 979, 987, 988, 1002, 1067,
    1083, 1084, 1097, 1581, 1597, 3204, 3219
\__enumext_tmp:nnn 585, 601, 602, 603, 604, 632, 648,
    649
\__enumext_tmp:nnnnnn 803, 828, 831, 834, 836, 838,
    841, 844
\__enumext_tmp:w . . . . . 4974, 4977
\l__enumext_tmpa_vii_int 4127, 4130, 4139, 4170
\l__enumext_tmpa_viii_int . . . . . 4158, 4161
\l__enumext_tmpa_X_dim . . . . . 172
\l__enumext_tmpa_X_int . . . . . 172
\l__enumext_topsep_v_skip . . 1156, 1160, 1360
\l__enumext_topsep_vii_skip . 1437, 1446, 1450
\l__enumext_topsep_viii_skip . 1459, 1481, 1485
\__enumext_undefine_anskey_env: . 78, 83, 2640,
    2640, 2891
\__enumext_unskip_unkern: . . 31, 236, 236, 1148,
    1177, 1210, 1382, 3626, 3627, 3645, 3766, 3767, 3784
\l__enumext_vspace_a_star_v_bool . . . . . 1630
\l__enumext_vspace_a_star_vii_bool . . 1652
\l__enumext_vspace_a_star_viii_bool . . 1663
\l__enumext_vspace_a_star_X_bool . . . . . 99
\__enumext_vspace_above: 57, 96, 1598, 1598, 3568
\__enumext_vspace_above_v: . 58, 1626, 1626, 3720
\l__enumext_vspace_above_v_skip . 1628, 1632,
    1634
\__enumext_vspace_above_vii: 58, 113, 1648, 1648,
    4489
\l__enumext_vspace_above_vii_skip 1650, 1654,
    1656
\__enumext_vspace_above_viii: . 58, 1648, 1659,
    4758
\l__enumext_vspace_above_viii_skip 1661, 1665,
    1667
\l__enumext_vspace_b_star_v_bool . . . . . 1641
\l__enumext_vspace_b_star_vii_bool . . 1674
\l__enumext_vspace_b_star_viii_bool . . 1685
\l__enumext_vspace_b_star_X_bool . . . . . 99
\__enumext_vspace_below: 57, 98, 1612, 1612, 3653
\__enumext_vspace_below_v: . 58, 1637, 1637, 3793
\l__enumext_vspace_below_v_skip . 1639, 1643,
    1645
\__enumext_vspace_below_vii: 59, 114, 1670, 1670,
    4499
\l__enumext_vspace_below_vii_skip 1672, 1676,
    1678
\__enumext_vspace_below_viii: . 59, 1670, 1681,
    4766
\l__enumext_vspace_below_viii_skip 1683, 1687,
    1689
\__enumext_widest_from:nnnn . 41, 761, 761, 776,
    795
\g__enumext_widest_label_tl 26, 36, 67, 500, 504,
    508
\l__enumext_wrap_label_opt_v_bool . . . 3290
\l__enumext_wrap_label_opt_vii_bool 115, 4552
\l__enumext_wrap_label_opt_viii_bool . 120,
    4797
\l__enumext_wrap_label_opt_X_bool . . . . . 99
\l__enumext_wrap_label_v_bool 3286, 3290, 3298,
    3336, 3351
\l__enumext_wrap_label_vii_bool . 115, 4552,
    4556, 4564, 4618
\l__enumext_wrap_label_viii_bool . 120, 4797,
    4801, 4809, 4881
\l__enumext_wrap_label_X_bool . . . . . 99
\__enumext_wrapper_label_v:n . 3338, 3353, 4081
\__enumext_wrapper_label_vii:n . . . . . 4623
\__enumext_wrapper_label_viii:n . . . . . 4883
\l__enumext_write_aux_file_tl . 29, 74, 84, 161,
    2447, 2453, 2962, 2968
enumext* . . . . . 5, 4434
enumXi . . . . . 472
enumXii . . . . . 472
enumXiii . . . . . 472
enumXiv . . . . . 472
enumXv . . . . . 472
enumXvi . . . . . 472

```

enumXvii 472

enumXviii 472

Environments provide by **enumext**:

 anskey* 28, 65, 73, 74, 76, 78, 79, 81, 83, 96, 114, 125, 130, 132

 enumext* .. 25, 26, 29-31, 34, 36, 39, 40, 42-47, 54, 55, 58-64, 66, 67, 69-78, 81, 83, 84, 89, 90, 94-96, 101, 107, 108, 111, 112, 114, 117-120, 122, 124-126, 128, 131, 134, 136

 enumext 25, 26, 30, 31, 34, 36-39, 41-50, 53, 55-57, 59-64, 66, 67, 69-78, 81, 83, 84, 87-90, 92-94, 96, 98-100, 103, 107, 110, 113, 114, 119, 124-126, 128, 131, 132, 134

 keyans* 25, 26, 28-32, 36, 39-42, 44-47, 54, 55, 58, 59, 65, 68, 69, 71, 79, 83, 89, 94, 101, 108, 109, 112, 119, 131, 133, 136

 keyanspic 25, 26, 28, 29, 32, 36, 37, 40, 65, 68, 71, 72, 79, 83-85, 101-106, 133

 keyans 25, 26, 28, 29, 31, 32, 36, 37, 40, 42, 44-47, 49, 53, 55-58, 65, 68, 69, 71, 72, 79, 83-85, 89-93, 98, 99, 102-104, 106, 110, 120, 131, 133

Environments:

 list 30, 33, 92, 94, 101, 103

 lrbox 117

 minipage 30, 34, 47, 50, 51, 102, 104, 105, 107, 110, 117, 123

 multicols 48-51, 56, 97, 98

 scontents 79, 81

exp commands:

 \exp_after:wN 4977

 \exp_args:Ne 2852, 2860, 3524, 4965

 \exp_args:NV ... 2572, 2727, 3230, 3248, 3270, 5249

 \exp_not:N . 58, 503, 620, 665, 682, 735, 936, 950, 951, 962, 963, 974, 975, 2512, 2543, 2544, 3001, 3066, 3067, 3079, 3080, 4853, 4854, 4974

 \exp_not:n 294, 309, 322, 330, 338, 559, 579, 620, 621, 665, 666, 682, 683, 735, 736, 937, 1719, 1728, 2179, 2276, 2288, 2450, 2478, 2488, 2498, 2512, 2513, 2819, 2832, 2842, 2965, 3003, 3005, 5078, 5088, 5281, 5286

F

\fbox 2162

\fboxrule 2162

\fboxsep 2162

file commands:

 \file_input_stop: 5685

first 988

font 512

\footnote 112

\footnote 112, 4409

\footnotemark 4419

\footnotesize 2544, 3067, 3080, 4854

\footnotetext 4403

\foreachkeyans 16, 128, 5214

G

\getkeyans 16, 124, 4963

group commands:

 \group_begin: .. 2542, 2587, 2762, 2849, 3065, 3078, 4852, 5020

 \group_end: 2549, 2603, 2866, 3072, 3085, 4859, 5027

H

\hbadness 4664, 4927

hbox commands:

 \hbox_overlap_left:n 3146, 4614

 \hbox_set:Nn 492, 3925

 \hbox_set_end: 4663, 4926

 \hbox_set_to_wd:Nnw 4637, 4892

\hfill 542, 547, 553, 554, 1537, 1564, 2512, 3001, 4304, 4368

hook commands:

 \hook_gput_code:nnn 9, 202, 206, 210, 409

 \hook_gremove_code:nn 81, 2778

 \hook_gset_rule:nnnn 410

 \hook_if_empty:nTF 2776

\hyperlink 75, 85

\hyperlink 2512, 3001

\hypertarget 35

\hypertarget 439

I

\IfDocumentMetadataTF 3163, 3324, 3821, 3829, 3837, 3871, 3879, 3887, 3948, 3958, 3966, 3976, 3981, 4007, 4015, 4045, 4054, 4302, 4366, 4446, 4454, 4643, 4666, 4710, 4718, 4898, 4929

\IfHyperBoolean 417

\IfPackageLoadedTF 11, 19, 413, 427

\ignorespaces 939, 4451, 4715

\inputlineno 294, 309, 322, 330, 338

int commands:

 \int_add:Nn 4212, 4261

 \int_case:nn ... 1112, 1230, 2027, 2053, 2092, 2116

 \int_case:nnTF 238

 \int_compare:nNnTF .. 395, 653, 670, 690, 697, 1200, 1219, 1373, 1391, 1503, 1519, 1531, 1559, 2140, 2146, 2611, 2615, 2619, 2627, 2673, 2677, 2681, 2878, 2899, 2938, 2943, 2948, 2973, 3061, 3506, 3517, 3539, 3552, 3590, 3605, 3620, 3639, 3705, 3709, 3737, 3762, 3778, 3896, 4028, 4032, 4182, 4192, 4208, 4231, 4241, 4257, 4464, 4468, 4507, 4517, 4673, 4685, 4732, 4744, 4936, 4948, 5132, 5264

 \int_compare_p:nNn ... 263, 273, 285, 286, 300, 301, 1509, 1510, 2033, 2059, 2395, 2405, 2417, 2418, 2433, 2474, 2650, 2651, 2662, 2663, 2815, 3549

 \int_decr:N 4211, 4260

 \int_eval:n .. 380, 790, 2294, 2445, 2544, 2956, 3067, 3080, 3406, 3450, 4200, 4249, 4854

 \int_from_alph:n 755, 769

 \int_from_roman:n 757, 771

 \int_gadd:Nn 4213, 4262

 \int_gdecr:N 2036, 2041, 2045, 2049, 2062

 \int_gincr:N 1869, 1874, 2457, 3011, 3100, 3134, 3304, 3581, 3729, 4070, 4531, 4600, 4776, 4843

 \int_gset:Nn 2085, 4417

 \int_gset_eq:NN 1768, 1775, 1781, 1787, 1795, 1802, 1808, 1814, 4414

 \int_gzero:N . 351, 352, 353, 1545, 1572, 2152, 2871, 3644, 3783, 4696, 4960

 \int_if_exist:NTF 1743, 1779, 1785, 1806, 1812, 1990

 \int_incr:N 2626, 3505, 3700, 3895, 4463, 4530, 4731, 4775

 \int_mod:nn 4687, 4950

 \int_new:N . 28, 29, 30, 31, 32, 33, 61, 62, 87, 103, 122, 142, 143, 154, 155, 156, 158, 169, 175, 176, 177, 178, 179, 1745, 1993

 \int_set:Nn 751, 755, 757, 1882, 1889, 1901, 1910, 2763, 4001, 4002, 4127, 4158, 4181, 4187, 4203, 4230, 4236, 4252, 4664, 4927, 5128, 5266

 \int_set_eq:NN 1870, 1875, 4210, 4259

 \int_sign:n 2087

 \int_step_function:nnN 2411, 2424, 2438

 \int_step_function:nnnN 5270

 \int_step_inline:nn 5180

`\int_step_inline:nnn` 4003
`\int_to_roman:n` 214, 2391, 2428
`\int_use:N` 373, 378, 379, 1201, 1220, 1532, 1884, 1891, 1903, 1912, 3406, 3425, 3450, 3525, 3591, 3600, 3615, 3621, 4185, 4186, 4198, 4234, 4235, 4247, 5601, 5605, 5611, 5615
`\int_zero:N` 4677, 4940
`\item` . 87, 90, 114, 117, 120, 122, 386, 2312, 2318, 2343, 2349, 2471, 2975, 2978, 3152, 3308, 3952, 3954, 4448, 4450, 4712, 4714, 4841
`\item*` 5, 14, 68, 3306
`item-pos*` 3204
`item-sym*` 3204
`\itemindent` 93
`\itemindent` 92
`itemindent` 883
`\itemsep` 3942
`\itemwidth` . 462, 2162, 3486, 3495, 3679, 3688, 4221, 4225, 4270, 4274

K

`keyans` 14, 3658
`keyans*` 14, 4700
`keyanspic` 15, 3944

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, 113
`above` 27, 57, 58, 96, 113, 119
`after` 46, 98, 114, 119
`align` 27, 37, 88, 91, 116, 130
`base-fix` 43, 59, 71, 95, 113, 125
`before*` 46, 96, 113, 119
`before` 46
`below*` 27, 57–59, 98, 114
`below` 27, 57–59, 98, 114, 119
`check-ans` 29, 30, 32, 64–68, 71, 82, 85, 98, 114, 118, 132
`columns-sep` 47, 97
`columns` 27, 47, 57, 97
`first` 46, 117
`font` 37, 88, 91, 116
`item-pos*` 87, 89
`item-sym*` 28, 87, 89
`itemindent` 27, 44, 87, 91, 117
`itemsep` 42, 94
`labelsep` 37, 93, 116
`labelwidth` 36–41, 93, 116
`label` 26, 36, 38, 41, 103, 107
`lisparindent` 94
`list-indent` 27, 44, 103
`list-offset` 44, 95, 99
`listparindent` 44, 117
`mark-ans` 69, 71, 76
`mark-pos` 69, 130
`mark-ref` 69, 71, 73, 75

`mini-env` 27, 34, 47, 56, 57, 71, 96, 110, 111, 113, 114, 119
`mini-right*` 27, 30, 47, 71, 110, 111, 113, 114
`mini-right` 27, 30, 47, 55, 71, 110, 111, 113, 114
`mini-sep` 27, 47, 71, 96
`no-store` 29, 64–66, 71, 77, 87
`noitemsep` 42
`nosep` 42
`parindent` 94
`parsep` 42, 94, 117
`partopsep` 42
`ref` 26, 30, 38–40, 131
`resume*` 26, 59, 60, 63–65, 71, 98, 114, 126
`resume` 26, 33, 59–65, 71, 98, 114, 126
`rightmargin` 44, 107
`save-ans` 28, 33, 60–64, 66, 67, 70–72, 77–79, 82–84, 90, 99, 105, 119–121, 124, 126, 132
`save-key` 28, 60, 70, 95, 113
`save-pos` 71
`save-ref` 29, 35, 69, 71, 73–75, 83, 85, 91, 121
`save-sep` 69, 71, 121
`series` 26, 59–63, 71, 95, 98, 113, 114, 126
`show-ans` 69, 71, 73, 74, 76, 91, 121
`show-length` 31, 45, 131
`show-pos` 28, 69, 73, 74, 76, 85, 91, 121
`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, 115, 116, 120
`wrap-label` 27, 37, 87, 88, 91, 103, 115, 116, 120
`wrap-opt` 69, 71

keys commands:

`\keys_define:nn` 514, 536, 571, 587, 634, 705, 779, 805, 849, 885, 908, 981, 990, 1069, 1086, 1583, 1694, 1937, 1998, 2157, 2199, 2235, 2240, 2554, 2705, 2741, 3206, 3222, 3242, 3262, 4991, 5090, 5206, 5214
`\keys_if_exist_p:nn` 5202, 5203
`\l_keys_key_str` 77, 80, 2572, 2727, 3230, 3248, 3270, 5249, 5357
`\keys_precompile:nnN` . 125, 198, 198, 4993, 4997, 5001, 5005, 5009, 5013, 5232
`\keys_set:nn` . 528, 865, 876, 1092, 1588, 1593, 1831, 1836, 1923, 1931, 2592, 3519, 3524, 3716, 4481, 4753, 5045, 5052, 5094, 5099, 5100, 5101, 5102, 5105, 5110, 5111, 5112, 5113, 5114, 5115, 5116, 5148, 5258
`\keys_set_known:nn` 2859

keyval commands:

`\keyval_parse:NNn` 1708, 2265, 5066

L

`label` 585, 632, 705
Labels provide by [enumext](#):
`\Alph*` 36
`\Roman*` 36
`\alph*` 36
`\arabic*` 30, 36
`\roman*` 36
`\labelsep` 3936, 3940
`labelsep` 512
`\labelwidth` 36
`\labelwidth` 3936, 3938
`labelwidth` 512
`\lastkern` 249

149 / 152

\prg_new_protected_conditional:Npnn ...	216
\prg_replicate:nn	233
\prg_return_false:	220
\prg_return_true:	219
\printkeyans	16, 124, <u>5018</u>
prop commands:	
\prop_const_from_keyval:Nn	5165
\prop_count:N 371, 2294, 2445, 2546, 2956, 3069, 3082,	4856, 5267
\prop_get:NnNTF	5191
\prop_gput_if_not_in:Nnn	2292
\prop_if_exist:NTF	1980, 4983, 5260
\prop_item:Nn	4985, 5284
\prop_new:N	1983
\ProvidesExplPackage	4
R	
\raggedcolumns	3614, 3756
\raisebox	4094
\ref	73, 83
ref	<u>585</u> , 632, 705
\refstepcounter	4597, 4871
regex commands:	
\regex_match:nnTF ..	218, 754, 756, 768, 770, 2791
\regex_replace_once:nnN	226
\renewcommand	620, 665, 682, 735
\RenewDocumentCommand 1549, 1576, 3152, 3171, 3187, 3308,	3332, 3346, 3954, 4409
\RequirePackage	17, 25
resume	<u>1692</u>
resume*	<u>1692</u>
rightmargin	<u>883</u>
\Roman	36, 41
\Roman	488
\roman	36, 41
\roman	489, 603, 5008
S	
\s	2792
save-ans	<u>1935</u>
save-key	<u>2233</u>
save-ref	<u>2155</u>
save-sep	<u>2155</u>
scan commands:	
\scan_stop:	3952, 4448, 4712, 4974, 4977
scontents internal commands:	
\l_scontents_fname_out_tl	2751
__scontents_parse_environment_keys:n .	2757
__scontents_rescan_tokens:n	2764
\l_scontents_storing_bool	2749
\l_scontents_writing_bool	2750
seq commands:	
\seq_clear:N	5126, 5269
\seq_const_from_clist:Nn	5119
\seq_count:N	372, 3973, 5130
\seq_gclear:N	4407, 4408
\seq_gput_right:Nn	2301, 4420, 4421
\seq_if_empty:NTF	4426, 5033, 5144
\seq_if_exist:NTF	1985, 5031
\seq_if_in:NnTF	5038
\seq_item:Nn	2789, 4013
\seq_map_function:NN	5135
\seq_map_inline:Nn	5046, 5053, 5145, 5146
\seq_map_pairwise_function:NNN	4428
\seq_new:N	123, 124, 126, 140, 170, 171, 1988
\seq_pop_left:NN	5134
\seq_put_right:Nn	4036, 5142, 5158, 5279
\seq_set_from_clist:Nn	5127
\seq_set_map_e:NNn	5136
\seq_show:N	5035
\seq_use:Nn	198, 199, 5275
series	<u>1692</u>
\setcounter	765, 769, 771, 3406, 3450, 3983
\setenumext	6, 126, <u>5124</u>
\setenumextmeta	6, 128, 5165
show-ans	<u>2155</u> , <u>2197</u>
show-length	<u>979</u>
show-pos	<u>2197</u>
skip commands:	
\skip_add:Nn 1117, 1123, 1129, 1139, 1143, 1168, 1172,	1189, 1247, 1249, 1263, 1266, 1287, 1289, 1303, 1306,
1326, 1328, 1342, 1345, 1364, 1413, 1414, 1425, 1427,	3930, 3937
\skip_gset:Nn	1440, 1444, 1448
\skip_gset_eq:NN	3934
\skip_gzero_new:N	1435, 1436
\skip_horizontal:N 951, 963, 975, 4615, 4632, 4680,	4887, 4943
\skip_horizontal:n ... 937, 2369, 2377, 3145, 3147,	4525, 4613, 4770, 4916
\skip_if_eq:nnTF 1115, 1121, 1127, 1233, 1273, 1313,	1401, 1437, 1459, 1600, 1614, 1628, 1639, 1650, 1661,
1672, 1683	
\skip_new:N ... 81, 82, 83, 88, 89, 90, 91, 92, 146, 190	
\skip_set:Nn 1100, 1104, 1154, 1158, 1183, 1236, 1237,	1255, 1276, 1277, 1295, 1315, 1316, 1334, 1358, 1404,
1405, 1419, 1439, 1443, 1461, 1465, 1469, 1475, 1479,	1483, 3914
\skip_set_eq:NN 1194, 1195, 1197, 1204, 1369, 1370,	1371, 1376, 3404, 3446, 3447, 4656, 4911
\skip_sub:Nn 1243, 1245, 1259, 1261, 1283, 1285, 1299,	1301, 1322, 1324, 1338, 1340, 1411, 1412, 1423, 1424
\skip_use:N 1102, 1106, 1141, 1145, 1150, 1170, 1174,	1185, 1191, 1601, 1605, 1608, 1615, 1619, 1622, 3628
\skip_vertical:N . 401, 404, 4318, 4332, 4695, 4959	
\skip_vertical:n	4694, 4958
\skip_zero:N 1203, 1217, 1355, 1356, 1357, 1375, 1389,	3448, 3611, 3753, 3941, 3942
\skip_zero_new:N	1434, 1456, 1457, 1458
\l_tmpa_skip 1255, 1265, 1268, 1295, 1305, 1308, 1334,	1344, 1347, 1419, 1426, 1428
\c_zero_skip . 401, 404, 1115, 1121, 1127, 1274, 1313,	1437, 1459, 1601, 1615, 1628, 1639, 1650, 1661, 1672,
1683, 4318, 4332, 4695, 4959	
\small	4996, 5000, 5004, 5008, 5012, 5016
socket commands:	
\socket_assign_plug:nn ..	3823, 3831, 3839, 3873,
3881, 3889	
\socket_new:nn	3795, 3843
\socket_new_plug:nnn 3796, 3803, 3811, 3844, 3851,	3860
\socket_use:n	3824, 3874
\socket_use:nn	3832, 3840, 3882, 3890
\star	3210
start	<u>777</u>
start*	<u>777</u>
start-list-tags	<u>3795</u> , <u>3843</u>
\stepcounter	3924, 4087, 4413
stop-list-tags	<u>3795</u> , <u>3843</u>

stop-start-tags 3795, 3843

str commands:

- `\c_backslash_str` 2637, 5322, 5327, 5332, 5337, 5339, 5341, 5346, 5348, 5446, 5450, 5454, 5464, 5468, 5476, 5477, 5481, 5493, 5494, 5498, 5499, 5520, 5522, 5526, 5528, 5556, 5619, 5621, 5625, 5627, 5636, 5637, 5641, 5646, 5647, 5651, 5655, 5659
- `\c_colon_str` 2444, 2955, 4974
- `\c_left_brace_str` 5427, 5434, 5440
- `\c_right_brace_str` 5427, 5434, 5440
- `\str_case:nn` 256, 315
- `\str_case:nnTF` . 1715, 1723, 2272, 2280, 5073, 5082
- `\str_clear:N` 3516, 4480
- `\str_count:n` 233
- `\str_if_empty:NTF` 1732, 1773, 1800
- `\str_if_eq:nnTF` 3407, 3452, 5175
- `\str_if_in:nnTF` 4970
- `\str_new:N` 84, 130, 145, 185
- `\str_set:Nn` . 543, 549, 555, 574, 575, 576, 2175, 2176, 2202, 2203, 3904, 3907
- `\str_use:N` 3191

`\string` 433

`\strutbox` . 1222, 1225, 1236, 1237, 1248, 1250, 1265, 1268, 1276, 1277, 1288, 1290, 1305, 1308, 1315, 1316, 1327, 1329, 1344, 1347, 1393, 1396, 1404, 1405, 1413, 1414, 1426, 1428, 1439, 1440, 1443, 1450, 1463, 1471, 1477, 1485, 3932, 3937, 3986, 4100

T

tag commands:

- `\tag_mc_begin:n` 3801, 3849, 3858
- `\tag_mc_end:` 3805, 3853, 3862
- `\tag_resume:n` . 3798, 3846, 3960, 3968, 4017, 4056, 4302, 4366
- `\tag_struct_begin:n` . 3799, 3800, 3807, 3808, 3809, 3847, 3848, 3855, 3856, 3857, 3969
- `\tag_struct_end:` 3981, 4454, 4718
- `\tag_struct_end:n` 3806, 3813, 3814, 3815, 3816, 3854, 3863, 3864, 3865, 3866
- `\tag_suspend:n` . 3817, 3867, 3950, 3962, 3978, 4009, 4047, 4446, 4710
- `\tag_tool:n` 3961

TeX and L^AT_EX 2_ε commands:

- `\@auxout` 449
- `\@currentenv` 256, 315
- `\protected@write` 449

tex commands:

- `\tex_newlinechar:D` 2763

text commands:

- `\text_expand:n` 4966
- `\textasteriskcentered` 2172, 2189
- `\the` 243, 249
- `\thepage` 455

tl commands:

- `\c_space_tl` 3037, 5371, 5386, 5409, 5413, 5600, 5601, 5610, 5611, 5671, 5675
- `\tl_clear:N` . 541, 548, 2153, 2219, 2229, 2250, 2258, 2464, 2783, 2784, 2898, 2972, 4816
- `\tl_clear_new:N` 498
- `\tl_const:Nn` 50, 482
- `\tl_gclear:N` . 363, 364, 365, 1753, 1758, 2873, 3182, 3200, 4338, 4398, 4616
- `\tl_gclear_new:N` 1740
- `\tl_gput_right:Nn` 483
- `\tl_greplace_all:Nnn` 504

- `\tl_gset:Nn` 291, 292, 306, 307, 1741, 1754, 1759, 1978, 2787, 3121, 4571
- `\tl_gset_eq:NN` 500, 3117, 4609
- `\tl_if_blank:nTF` 2576, 2594, 2731, 3234, 3252, 3274, 4607, 5239
- `\tl_if_empty:NTF` . 608, 627, 655, 672, 692, 699, 725, 742, 1766, 1771, 1793, 1798, 1856, 1920, 1928, 1957, 2017, 2308, 2339, 2484, 2828, 2850, 2880, 2909, 2982, 3031, 3142, 4819, 5156
- `\tl_if_empty:nTF` 1821
- `\tl_if_exist:NTF` 1826
- `\tl_if_novalue:nTF` . 2590, 2906, 2980, 3016, 3096, 3115, 3123, 3284, 3514, 3971, 4411, 4478, 4751, 4817
- `\tl_map_inline:Nn` 224, 501
- `\tl_new:N` 42, 43, 44, 47, 52, 53, 56, 57, 63, 65, 66, 68, 69, 104, 105, 106, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 125, 127, 128, 129, 131, 134, 135, 153, 161, 162, 163, 166, 184
- `\tl_put_left::Ne` 2817
- `\tl_put_left:Nn` 2316, 2347, 2469, 2811, 2824, 2830, 2840, 3048, 3088, 4322, 4383, 4838, 4841
- `\tl_put_right:Nn` 499, 618, 663, 680, 733, 2320, 2351, 2398, 2408, 2421, 2436, 2442, 2447, 2471, 2476, 2483, 2486, 2496, 2501, 2504, 2510, 2901, 2904, 2911, 2913, 2940, 2945, 2950, 2953, 2962, 2975, 2978, 2984, 2989, 2999, 4821, 4825
- `\tl_remove_all:Nn` 5155
- `\tl_remove_once:Nn` 2386, 2925
- `\tl_replace_all:Nnn` 503, 5190
- `\tl_reverse:N` 2385, 2387, 2924, 2926
- `\tl_set:Nn` . 58, 260, 270, 319, 320, 327, 328, 335, 336, 468, 542, 547, 553, 554, 607, 652, 724, 934, 948, 960, 972, 1855, 1956, 2220, 2230, 2251, 2259, 2540, 2751, 3018, 3063, 3076, 4827, 4850, 5153, 5189, 5259
- `\tl_set_eq:NN` 509, 613, 616, 660, 662, 677, 679, 730, 732, 2384, 2923, 2936, 3296, 3300, 4075, 4077
- `\tl_to_str:n` 1826, 1832, 1837, 4966
- `\tl_trim_spaces:n` . 499, 5142, 5153, 5159, 5175
- `\tl_use:N` 505, 508, 629, 694, 701, 744, 1005, 1009, 1013, 1017, 1021, 1025, 1029, 1033, 1037, 1041, 1045, 1049, 1053, 1057, 1061, 1065, 2374, 2391, 2399, 2410, 2423, 2428, 2439, 3104, 3110, 3138, 3173, 3174, 3181, 3193, 3287, 3291, 3299, 3334, 3335, 3341, 3350, 3473, 3664, 4080, 4329, 4390, 4622, 4629, 4654, 4657, 4880, 4909, 4914, 4919, 5021, 5022, 5023, 5024, 5025, 5043, 5138, 5257

token commands:

- `\token_to_str:N` 451
- `\topsep` 3937
- `topsep` 803
- `\topskip` 1203, 1375
- `\typeout` 242, 243, 248, 249, 419, 422, 432, 433

U

- `\u` 227, 2792
- `\unkern` 250
- `unknown` 3220, 3242, 3260
- `\unskip` 244

use commands:

- `\use:N` 234, 3178, 3197, 3475
- `\use:n` 1706, 2263, 4972, 5064
- `\use_none:nn` 443, 5196
- `\usecounter` 3405, 3449

V

- `\value` 1769, 1775, 1782, 1788, 1796, 1802, 1809, 1815

vbox commands:		wrap-ans	<u>2155</u>
\ vbox_set:Nn	4049	wrap-label	<u>512</u>
\ vbox_set_top:Nn	4327, 4388	wrap-label*	<u>512</u>
\ vspace . 864, 875, 1605, 1608, 1619, 1622, 1632, 1634, 1643,		wrap-opt	<u>2155</u>
1645, 1654, 1656, 1665, 1667, 1676, 1678, 1687, 1689			
W		Z	
widest	<u>777</u>	\ z	2792