

enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-06-29^{*}

©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	130
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 2024-06-29.

[†]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 contents[4] packages, need to have a modern TeX distribution such as TeX Live or MiKTeX. It has been tested with the standard classes provided by LaTeX: book, report, article and letter on 10pt, 11pt and 12pt.

1 Introduction

In the LaTeX 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) LaTeXze 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) LaTeXze 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

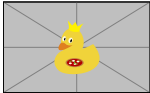
B)

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 / 144

1. B), $x = 5$

2. D)

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

* 5. D), “other note”

*

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

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

1.1 Description and usage

The `enumext` package defines enumerated environments using the `list` environment provided by \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 $\text{MiK}\text{\TeX}$, use the package manager to install. For manual installation, download `enumext.zip` and unzip it, run `lualatex enumext.dtx` and move all files to appropriate locations, then run `mktexlsr`. To produce the documentation run `lualatex enumext.dtx` two times.

enumext.sty

enumext.pdf

README.md

enumext.dtx

»

TDS:tex/latex/enumext/

TDS:doc/latex/enumext/

TDS:doc/latex/enumext/

TDS:source/latex/enumext/

The package is loaded in the usual way:

\usepackage{enumext}

1.2 The concept of left margin

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

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

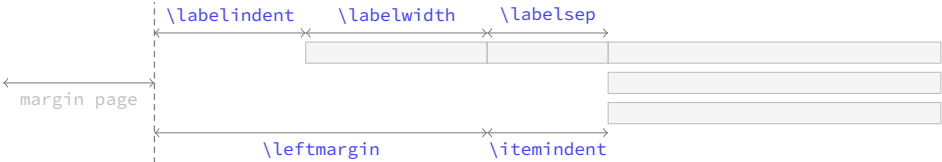


Figure 1: Representation of horizontal lengths in `enumitem`.

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

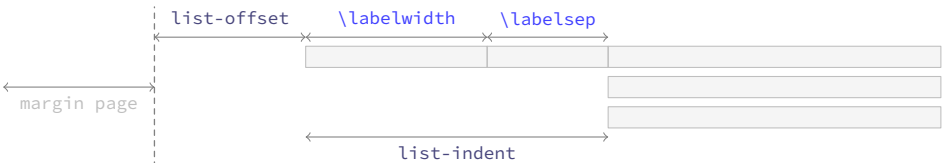


Figure 2: Representation of horizontal lengths concept in `enumext`.

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

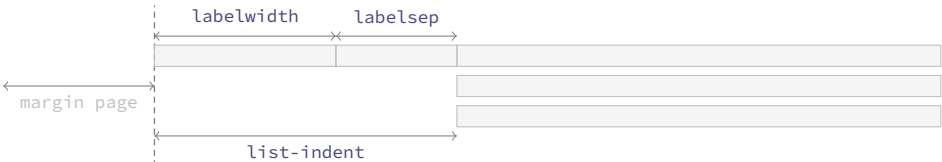


Figure 3: Default horizontal lengths `list-offset=0pt`, `list-indent=\labelwidth+\labelsep` in `enumext`.

1.3 User interface

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

1.3.1 Internal counters

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

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

1.3.2 Public dimension

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

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

1.3.3 Support for multicol

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



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

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

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

1.3.4 Support for minipage

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



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

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

1.3.5 The \label and \ref system

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

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

1.3.6 Support for \footnote

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

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

2 The environments provided

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

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

2.1 The environment enumext

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

Example with columns=2

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

2.2 The environment enumext*

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

Some considerations to take into account for this environment:

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

Example with columns=2

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

2.3 The command \item*

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

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

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

2.3.1 Keys for `\item*`

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

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

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

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

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

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

`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 discardable”.

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

default: *not used*

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

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

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

5.2.2 Horizontal spaces

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

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

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

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

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

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

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

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

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

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

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

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

5.3 Keys for add code

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

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

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

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

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

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

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

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

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

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

5.4 Keys for start, series and resume

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

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

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

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

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

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

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

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

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

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

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

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

5.5 Keys for multicols

`columns = {⟨integer⟩}` default: `1`

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

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

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

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

5.6 Keys for minipage

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

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

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

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

5.6.1 The command `\miniright`

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

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

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

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

5.6.2 The key `mini-right`

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

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

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

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

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

6 The storage system

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

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

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

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

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

6.1 Keys for storage system

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

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

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

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

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

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

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

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

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

6.1.1 Keys for label and ref

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

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

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

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

6.1.2 Keys for wrap and display

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

6.1.3 Keys for debug and checking

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

6.2 The command `\anskey`

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

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

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

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

6.2.1 Keys for `\anskey`

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

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

`item-sym*` = { $\langle symbol \rangle$ } default: $\$star\$$
 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*}

```

```

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

```

* 5. Text containing our instructions or questions.

[5] First answer with verbatim

6. Text containing our instructions or questions.

(a) Question.

[6] second answer

7. Text containing our instructions or questions.

[7] third answer

8. Text containing our instructions or questions.

[8] fourth answer

6.4 The environments `keyans` and `keyans*`

```

keyans \begin{keyans}[\langle key = val \rangle] \item \item[\langle custom \rangle] \item* \item*[\langle content \rangle] \end{keyans}
keyans* \begin{keyans*}[\langle key = val \rangle] \item \item[\langle custom \rangle] \item* \item*[\langle content \rangle] \end{keyans*}

```

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

```

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

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

```

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

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

```

\item* \item*
\item*[\langle content \rangle]

```

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

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

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

Example

```

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

```




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

1. Text containing a question.

A) Choice
C) Choice
E) Choice

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

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


Some text

6.5 The environment keyanspic

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

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

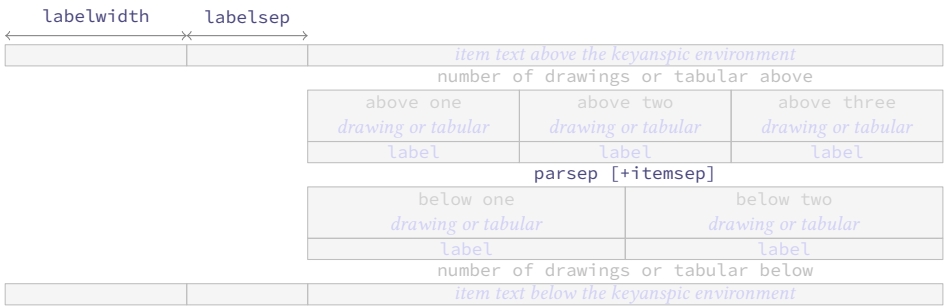


Figure 6: Representation of the `keyanspic` environment with optional argument [3,2] in `enumext`.

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

6.5.1 The command \anspic

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

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

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

Example

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

1. Question with images.



A)



B)



C)



D)



* E)[note]

6.6 Printing stored content

6.6.1 The command `\getkeyans`

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

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

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

6.6.2 The command `\foreachkeyans`

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

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

Options for command

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

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

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

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

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

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

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

6.6.3 The command `\printkeyans`

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

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

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

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

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

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

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

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

Example

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

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

  \item Related to Linux

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

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

```
\printkeyans{sample}
```

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

[1]

2. True False

(a)

[2]

3. Related to Linux

(a) You use linux?

[3]

(b) Rate the following package and class

i.

[4]

ii.

[5]

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

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

2. (a) Very True!

3. (a) Yes

(b) i. very good

ii. obsolete
- *

*

*

*

*


7 Full examples

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

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


and then you can use the excellent [arara](#)¹ tool to compile them.

Example 1

Adapted from the response given by Enrico Gregorio in [Squares for answer choice options and perfect alignment to mathematical answers](#) 

1. La velocità di $1,00 \times 10^2$ m/s espressa in km/h è:
 - ☐ A 36 km/h.
 - ☐ B 360 km/h.
 - ☐ C 27,8 km/h.
 - ☐ D $3,60 \times 10^8$ km/h.
 2. In fisica nucleare si usa l'angstrom (simbolo: $1 \text{ \AA} = 1 \times 10^{-10}$ m) e il fermi o femtometro ($1 \text{ fm} = 1 \times 10^{-15}$ m). Qual è la relazione tra queste due unità di misura?
 - ☐ A $1 \text{ \AA} = 1 \times 10^5 \text{ fm}$.
 - ☐ B $1 \text{ \AA} = 1 \times 10^{-5} \text{ fm}$.
 - ☐ C $1 \text{ \AA} = 1 \times 10^{-15} \text{ fm}$.
 - ☐ D $1 \text{ \AA} = 1 \times 10^3 \text{ fm}$.
 3. La velocità di $1,00 \times 10^2$ m/s espressa in km/h è:
 - ☐ A 36 km/h.
 - ☐ B 360 km/h.
 - ☐ C 27,8 km/h.
 - ☐ D $3,60 \times 10^8$ km/h.
 4. In fisica nucleare si usa l'angstrom (simbolo: $1 \text{ \AA} = 1 \times 10^{-10}$ m) e il fermi o femtometro ($1 \text{ fm} = 1 \times 10^{-15}$ m). Qual è la relazione tra queste due unità di misura?
 - ☐ A $1 \text{ \AA} = 1 \times 10^5 \text{ fm}$.
 - ☐ B $1 \text{ \AA} = 1 \times 10^{-5} \text{ fm}$.
 - ☐ C $1 \text{ \AA} = 1 \times 10^{-15} \text{ fm}$.
 - ☐ D $1 \text{ \AA} = 1 \times 10^3 \text{ fm}$.

Example 2

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

1. La velocità di $1,00 \times 10^2$ m/s espressa in km/h è:
- ☐ A 36 km/h.
- ☒ B 360 km/h.
- ☐ C 27,8 km/h.
- ☐ D $3,60 \times 10^8$ km/h.
2. In fisica nucleare si usa l'angstrom (simbolo: $1 \text{ \AA} = 1 \times 10^{-10}$ m) e il fermi o femtometro ($1 \text{ fm} = 1 \times 10^{-15}$ m). Qual è la relazione tra queste due unità di misura?
- ☒ A $1 \text{ \AA} = 1 \times 10^5 \text{ fm}$.
- ☐ B $1 \text{ \AA} = 1 \times 10^{-5} \text{ fm}$.
- ☐ C $1 \text{ \AA} = 1 \times 10^{-15} \text{ fm}$.
- ☐ D $1 \text{ \AA} = 1 \times 10^3 \text{ fm}$.
3. La velocità di $1,00 \times 10^2$ m/s espressa in km/h è:
- ☐ A 36 km/h.
- ☒ B 360 km/h.
- ☐ C 27,8 km/h.
- ☐ D $3,60 \times 10^8$ km/h.
4. In fisica nucleare si usa l'angstrom (simbolo: $1 \text{ \AA} = 1 \times 10^{-10}$ m) e il fermi o femtometro ($1 \text{ fm} = 1 \times 10^{-15}$ m). Qual è la relazione tra queste due unità di misura?
- ☒ A $1 \text{ \AA} = 1 \times 10^5 \text{ fm}$.
- ☐ B $1 \text{ \AA} = 1 \times 10^{-5} \text{ fm}$.
- ☐ C $1 \text{ \AA} = 1 \times 10^{-15} \text{ fm}$.
- ☐ D $1 \text{ \AA} = 1 \times 10^3 \text{ fm}$.
1. B
2. A
3. B
4. A

Example 3

A “simple multiple choice” test

1. First type of questions
 - (A) value
 - (B) correct
 - (C) value
 - (D) value
2. Second type of questions
 - I. $2\alpha + 2\delta = 90^\circ$

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

- II. $\alpha = \delta$

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

A I only

B II only

C I and II only
- D I and III only

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

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

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

A value

B value

C value

D value

E value

4. Question with image and label below:
-
5. Question with image on left side:

A value

B value

C value

D correct

E value
- Test keys
1. B, $x = 5$

2. D

3. C, some note

* 4. E, A duck

* 5. D, other note

*
- Example 4
- A “simple worksheet” using ducks :) 🦆.
- Factor $x^2 - 2x + 1$
- Factor $3x + 3y + 3z$
- The following questions need to be cuaqtified :)
- True False
- (a) $\alpha > \delta$

(b) ~~ETX~~e is cool?
- Related to Linux
- (a) You use linux?

(b) Usually uses the package manager?

(c) Rate the following package and class

i. `xsim-exam`

ii. `xsim`

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

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

3. (a) False

(b) Very True!

4. (a) Yes

* (b) Yes, dnf

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

* ii. very good

* iii. obsolete

*
- Example 5
- Adapted from the response given by Stephen in SAT like question format 📄.
- 1

Which choice best describes what happens in the passage?

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

B) One character receives a surprising request

from another character.

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

D) One character criticizes another character for pursuing an unexpected course of action.

2
- ©2024 by Pablo González L
- 19 / 144

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

- SomeThing

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

A short one-line description.
- Something long

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

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

- SomeThing

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

A short one-line description.
- Something long

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

If we add `align=right` it will look like this:

- SomeThing

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

A short one-line description.
- Something long

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

At this point we have used `list-offset={-\descitemwd - 4pt}` instead of `list-offset={-\labelwidth - \labelsep}`, this is because the parameters `\labelwidth` and `\labelsep` take the default values, as if we had not set `label`.

Description with multi-line labels

The `label` key does not accept *multiline material*, this is where the `wrap-label*` key comes into play. Unlike the `enumitem` package, the `align` key only supports three options, so what we will do is create a command in the style `\parleft` of `enumitem` that allows us to place *multiline labels* using `\parbox`.

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

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

- SomeThing

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

A short one-line description.
- Something long

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

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

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. “ \LaTeX 2_ε: 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 \LaTeX 3 Interfaces”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2024.
- [12] The \LaTeX Project. “The \LaTeX 2_ε 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 2024-06-29 – First public release.

11 Index of Documentation

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

C

Document class:

article 2

book 2

exam 2

letter 2

report 2

\columnbreak 4, 12

\columnsep 10

Commands provide by enumext:

\anskey 11–13

\anspic 11, 12, 15

\foreachkeyans 16

\getkeyans 12, 16

\item* 5–7, 11, 12, 14, 15

\item 5–10, 12, 14

\miniright 10

\printkeyans 6, 11, 16

\setenumextmeta 6

\setenumext 5–7, 11, 12, 14, 17

Counters defined by enumext:

enumXiii 4

enumXii 4

enumXiv 4

enumXi 4

enumXviii 4

enumXvii 4

enumXvi 4

enumXv 4

E

Environments provide by enumext:

anskey* 11–13

enumext* 4–14, 16, 17

enumext 4–14, 16, 17, 20

keyans* 4–14

keyanspic 4, 7, 8, 11–13, 15, 20

keyans 4–9, 11–15, 20

Environments:

Verbatim 13

enumerate 1, 3, 5, 21

figure 5

list 3, 9, 22

minipage 3–5, 10, 22

multicols 3, 4, 10

table 5

task 5

F

\footnote 5

I

\itemsep 8

K

Keys for \anskey provide by enumext:

break-col 12

item-join 12

item-pos* 13

item-star 12, 13

item-sym* 13

Keys for \foreachkeyans provide by enumext:

after 16

before 16

sep 16

start 16

step 16

stop 16

wrapper 16

Keys for anskey* provide by enumext:

break-col 12

force-eol 13

item-join 12

item-pos* 13

item-star 12, 13

item-sym* 13

overwrite 13

write-env 13

Keys for environments provide by enumext:

above* 8

above 8

after 9, 10

align 7, 21

base-fix 8

before* 9

before 9

below* 9

below 8

check-ans 12

columns-sep 4, 10

columns 4, 8, 10

first 9

font 7

item-pos* 5, 6

item-sym* 5, 6

itemindent 9

itemsep 8, 15

labelsep 3–7, 9, 10, 12, 20, 21

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

labelwith 5

label 7, 9, 14, 20, 21

list-indent 3, 9

list-offset 3, 9, 21

listparindent 9

mark-ans 12

mark-pos 12

mark-ref 11

mini-env 4, 5, 8, 10

mini-right* 7, 10

mini-right 7, 10

mini-sep 4, 10

no-store 11–13

noitemsep 8

nosep 8, 20

overwrite 13

parsep 8, 15

partopsep 8

ref 4, 7

resume* 7, 10, 11

resume 7, 10, 11

rightmargin 9

save-ans 4, 6, 10–16

save-key	10, 11, 17	\linewidth	10
save-ref	4, 7, 11-13, 16	\listparindent	9
save-sep	11		
series	7, 10, 11	P	
show-ans	11, 12	Packages:	
show-length	8	enumerate	21
show-pos	11, 12, 16	enumext	1-5, 7, 15, 21
start*	9, 10	enumitem	3-5, 9, 21
start	9, 10	fancyvrb	13
topsep	8, 9	footnotehyper	5
widest	7	hyperref	4, 5, 11-13, 21, 22
wrap-ans	12	l3keys	7
wrap-label*	8, 21	l3prop	1, 21
wrap-label	7, 8	l3seq	1, 21
wrap-opt	12	multicol	1, 2, 4, 22
write-env	13	scontents	1, 2, 13
		task	5, 6
L		xsim	2
\label	4	\parsep	8
Labels provide by enumext:		\partopsep	8
\Alph*	7, 14		
\Roman*	7	R	
\alph*	7	\raggedcolumns	4
\arabic*	7	\ref	4
\roman*	7	\rightmargin	9
\labelsep	3, 7		
\labelwidth	3, 7	T	
		\topsep	8

12 Implementation

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

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

12.1 General conventions

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

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

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

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

12.2 Initial set up

Start the DocStrip guards.

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

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

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

12.3 Declaration of the package

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

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

Now declare the `enumext` package.

```
4 \ProvidesExplPackage
5   {enumext}
6   {2024-06-29}
7   {1.0}
8   {Enumerate exercise sheets}
```

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

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

12.4 Definition of variables

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

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

```

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

```

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

```

\c__enumext_counter_style_tl
\l__enumext_ref_key_arg_tl
\l__enumext_ref_the_count_tl
\l__enumext_the_counter_X_tl
\l__enumext_renew_the_count_X_tl

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

\l__enumext_current_widest_dim
\g__enumext_counter_styles_tl
\g__enumext_widest_label_tl
\l__enumext_label_width_by_box

```

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

```

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

```


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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```

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

```

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

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

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

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

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

```

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

```

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

```

\l__enumext_setkey_tmpa_tl
\l__enumext_setkey_tmpb_tl
\l__enumext_setkey_tmpa_int
\l__enumext_setkey_tmpa_seq
\l__enumext_setkey_tmpb_seq

```

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

```

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

```

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

```

\l__enumext_meta_path_tl
\l__enumext_foreach_print_seq
\l__enumext_foreach_name_prop_tl
\l__enumext_foreach_default_keys_tl

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

\l__enumext_keyans_pic_body_seq
\l__enumext_keyans_pic_width_dim
\l__enumext_keyans_pic_above_int
\l__enumext_keyans_pic_below_int
\l__enumext_keyans_pic_above_skip

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

\l__enumext_hyperref_bool
\l__enumext_footnotes_key_bool

```

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

```

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

```

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

```

\l__enumext_newlabel_arg_one_tl
\l__enumext_newlabel_arg_two_tl
\l__enumext_write_aux_file_tl
\l__enumext_label_copy_X_tl

```

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

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

```

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

```

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

```

\g__enumext_footnote_int
\g__enumext_footnote_arg_seq
\g__enumext_footnote_int_seq

```

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

```

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

```

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

```

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

```

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

```

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

```

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

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

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

(End of definition for `\c__enumext_all_envs_clist`.)

12.5 Some utility functions

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

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

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

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

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

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

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

`__enumext_before_env:nn`

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

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

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

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

(End of definition for `__enumext_level:`.)

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

`__enumext_if_is_int:nF`

`__enumext_if_is_int:nTF`

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

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

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

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

(End of definition for `__enumext_regex_counter_style:`.)

`__enumext_show_length:nnn`

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

```

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

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

12.5.1 Utilities for environments and levels

`__enumext_is_not_nested:`
`__enumext_is_on_first_level:`

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

```

229 \cs_new_protected:Nn \__enumext_is_not_nested:
230 {
231     \str_case:en { \@currentenvir }
232     {
233         {enumext}
234         {
235             \tl_set:Nn \l__enumext_envir_name_tl { enumext }
236             \bool_lazy_and:nnT
237             { \bool_not_p:n { \g__enumext_standar_bool } }
238             { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
239             {
240                 \bool_gset_true:N \g__enumext_standar_bool
241             }
242         }
243         {enumext*}
244         {
245             \tl_set:Nn \l__enumext_envir_name_tl { enumext* }
246             \bool_lazy_and:nnT
247             { \bool_not_p:n { \g__enumext_starred_bool } }
248             { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
249             {
250                 \bool_gset_true:N \g__enumext_starred_bool
251             }
252         }
253     }
254 }
```

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

```

255 \cs_new_protected:Nn \__enumext_is_on_first_level:
256 {
257     \bool_lazy_all:nT
258     {
259         { \bool_if_p:N \g__enumext_standar_bool }
260         { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
261         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
262     }
263     {
264         \bool_set_true:N \l__enumext_standar_first_bool
265         \bool_set_true:N \l__enumext_anskey_env_bool
266         \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
267         \tl_gset:Nn \g__enumext_start_line_tl
268         {
269             on ~ line ~ \exp_not:V \inputlineno
270         }
271     }
272     \bool_lazy_all:nT
273     {
274         { \bool_if_p:N \g__enumext_starred_bool }
275         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
```

```

276     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
277   }
278   {
279     \bool_set_true:N \l__enumext_starred_first_bool
280     \bool_set_true:N \l__enumext_anskey_env_bool
281     \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
282     \tl_gset:Ne \g__enumext_start_line_tl
283       {
284         on ~ line ~ \exp_not:V \inputlineno
285       }
286   }
287 }

```

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

`__enumext_keyans_name_and_start:`

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

```

288 \cs_new_protected:Nn \__enumext_keyans_name_and_start:
289 {
290   \str_case:en { \@currenvir }
291   {
292     {keyans}
293     {
294       \tl_set:Nn \l__enumext_envir_name_tl { keyans }
295       \tl_set:Ne \l__enumext_check_start_line_env_tl
296         {
297           in ~ 'keyans' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
298         }
299     }
300     {keyans*}
301     {
302       \tl_set:Nn \l__enumext_envir_name_tl { keyans* }
303       \tl_set:Ne \l__enumext_check_start_line_env_tl
304         {
305           in ~ 'keyans*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
306         }
307     }
308     {keyanspic}
309     {
310       \tl_set:Nn \l__enumext_envir_name_tl { keyanspic }
311       \tl_set:Ne \l__enumext_check_start_line_env_tl
312         {
313           in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
314         }
315     }
316   }
317 }

```

(End of definition for `__enumext_keyans_name_and_start:`)

12.5.2 Utilities for log and terminal

`__enumext_reset_global_vars:`

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

`__enumext_reset_global_int:`

`__enumext_reset_global_bool:`

`__enumext_reset_global_tl:`

```

318 \cs_new_protected:Nn \__enumext_reset_global_vars:
319 {
320   \__enumext_reset_global_int:
321   \__enumext_reset_global_bool:
322   \__enumext_reset_global_tl:
323 }
324 \cs_new_protected:Nn \__enumext_reset_global_int:
325 {
326   \int_gzero:N \g__enumext_item_number_int
327   \int_gzero:N \g__enumext_item_anskey_int
328   \int_gzero:N \g__enumext_item_answer_diff_int
329 }
330 \cs_new_protected:Nn \__enumext_reset_global_bool:
331 {
332   \bool_gset_false:N \g__enumext_check_ans_key_bool
333   \bool_gset_false:N \g__enumext_standar_bool
334   \bool_gset_false:N \g__enumext_starred_bool

```



```

335   }
336   \cs_new_protected:Nn \__enumext_reset_global_tl:
337   {
338     \tl_gclear:N \g__enumext_store_name_tl
339     \tl_gclear:N \g__enumext_start_line_tl
340     \tl_gclear:N \g__enumext_envir_name_tl
341   }

```

(End of definition for __enumext_reset_global_vars: and others.)

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

```

342   \cs_new_protected:Nn \__enumext_log_global_vars:
343   {
344     \msg_log:nneeee { enumext } { prop-seq-int-hook }
345     { \g__enumext_store_name_tl }
346     { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
347     { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
348     { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
349   }

```

The function __enumext_log_answer_vars: will be passed to the function __enumext_execute_after_env: and write to the .log file the number of items and answers along with the difference between them.

```

350   \cs_new_protected:Nn \__enumext_log_answer_vars:
351   {
352     \msg_log:nneeee { enumext } { item-answer-hook }
353     { \int_use:N \g__enumext_item_number_int }
354     { \int_use:N \g__enumext_item_anskey_int }
355     { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int } }
356   }

```

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

12.6 Copying list and minipage environments

The *list* environment provided by L^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.

```

357   \__enumext_at_begin_document:n
358   {
359     \cs_new_eq:NN \__enumext_start_list:nn \list
360     \cs_new_eq:NN \__enumext_stop_list: \endlist
361     \cs_new_eq:NN \__enumext_item_std:w \item
362   }

```

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

The *minipage* environment provided by L^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.

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

```

363   \__enumext_at_begin_document:n
364   {
365     \cs_new_eq:NN \__enumext_minipage:w \minipage
366     \cs_new_eq:NN \__enumext_endminipage: \endminipage
367   }

```

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

12.7 The internal minipage environment

```
\__enumext_internal_mini_page:
__enumext_mini_env*
```

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

```
368 \cs_new_protected:Nn \__enumext_internal_mini_page:
369 {
370   \int_compare:nNtT { \l__enumext_level_int } = { 0 }
371   {
372     \DeclareDocumentEnvironment{\__enumext_mini_env*}{ m }
373     {
374       \__enumext_minipage:w [ t ] { ##1 }
375       \legacy_if_gset_false:n { @minipage }
376       \skip_vertical:N \c_zero_skip
377     }
378     {
379       \skip_vertical:N \c_zero_skip
380       \__enumext_endminipage:
381     }
382   }
383 }
```

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

12.8 Compatibility with hyperref and footnotehyper

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

```
384 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
385 \hook_gset_rule:nnnn { begindocument } { enumext } { after } { hyperref }
```

```
\__enumext_after_hyperref:
__enumext_hypertarget:nn
__enumext_phantomsection:
```

The function `__enumext_after_hyperref:` sets the state of the boolean variable `\l__enumext_hyprerref_bool` to “*true*” if the package is loaded. At this point we will use the public macro `\IfHyperBoolean` to determine if the `hyperfootnotes=true` key is present, if so, we set the state of the boolean variable `__enumext_footnotes_key_bool` to “*true*”.

```
386 \cs_new_protected:Nn \__enumext_after_hyperref:
387 {
388   \IfPackageLoadedTF { hyperref }
389   {
390     \msg_info:nnn { enumext } { package-load } { hyperref }
391     \bool_set_true:N \l__enumext_hyprerref_bool
392     \IfHyperBoolean{hyperfootnotes}
393     {
394       \typeout{hyperfootnotes=true}
395       \bool_set_true:N \l__enumext_footnotes_key_bool
396     }
397     { \typeout{hyperfootnotes=false} }
398   }
399   { }
```

If the state of the variable `\l__enumext_footnotes_key_bool` is true we will check if the package `footnotehyper` is loaded, in case it is not present, we will set the value of `\l__enumext_footnotes_key_bool` to false and we will redefine `\footnote`.

```
400   \bool_if:NT \l__enumext_footnotes_key_bool
401   {
402     \IfPackageLoadedTF { footnotehyper }
403     {
404       \msg_info:nnn { enumext } { package-load } { footnotehyper }
405     }
406     {
407       \typeout{No ~ footnotehyper ~ load}
408       \typeout{Load ~ and ~ use ~ \string\makesavenoteenv{enumext*}}
409       \bool_set_false:N \l__enumext_footnotes_key_bool
410     }
411   }
```

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

```

412 \bool_if:NTF \__enumext_hyperref_bool
413 {
414   \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
415   \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
416 }
417 {
418   \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
419   \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
420 }
421 }
```

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

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

#1: `__enumext_newlabel_arg_one_tl`
 #2: `__enumext_newlabel_arg_two_tl`

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

```

422 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
423 {
424   \protected@write \auxout { }
425   {
426     \token_to_str:N \newlabel {#1}
427     {
428       {#2}
429       \bool_if:NT \__enumext_hyperref_bool
430       { { \thepage } {#2} {#1} }
431       { }
432     }
433   }
434   \__enumext_hypertarget:nn {#1} { }
435   \__enumext_phantomsection:
436 }
```

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

12.9 Definition of public dimension

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

```

437 \dim_zero_new:N \itemwidth
```

12.10 Definition of counters

`__enumext_define_counters:Nn`
`__enumext_define_counters:cn`

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

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

```

438 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
439 {
440   \cs_if_exist:cTF { c@ #2 }
441   { \msg_fatal:nnn { enumext } { counters } { #2 } }
442   {
443     \tl_set:Nn #1 { #2 }
444     \newcounter { #2 }
445   }
446 }
```

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

```

enumXi    The counters created here are enumXi, enumXii, enumXiii and enumXiv for enumext environment, enumXv
enumXii   for keyans environment, enumXvi for keyanspic environment, enumXvii for enumext* and enumXviii
enumXiii  for the keyans* environments.
enumXiv   447 \__enumext_define_counters:Nn \__enumext_counter_i_tl { enumXi }
enumXv    448 \__enumext_define_counters:Nn \__enumext_counter_ii_tl { enumXii }
enumXvi   449 \__enumext_define_counters:Nn \__enumext_counter_iii_tl { enumXiii }
enumXvii  450 \__enumext_define_counters:Nn \__enumext_counter_iv_tl { enumXiv }
enumXviii 451 \__enumext_define_counters:Nn \__enumext_counter_v_tl { enumXv }
          452 \__enumext_define_counters:Nn \__enumext_counter_vi_tl { enumXvi }
          453 \__enumext_define_counters:Nn \__enumext_counter_vii_tl { enumXvii }
          454 \__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.

```

455 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
456 {
457   \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
458   \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
459 }
460 \__enumext_register_counter_style:Nn \arabic { 0 }
461 \__enumext_register_counter_style:Nn \Alph { M }
462 \__enumext_register_counter_style:Nn \alph { m }
463 \__enumext_register_counter_style:Nn \Roman { VIII }
464 \__enumext_register_counter_style:Nn \roman { viii }

```

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

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

`__enumext_label_width_by_box:cv`

```

465 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
466 {
467   \hbox_set:Nn \l__enumext_label_width_by_box {#2}
468   \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
469 }
470 \cs_generate_variant:Nn \__enumext_label_width_by_box:Nn { cv }

```

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

`__enumext_label_style:Nnn` The function `__enumext_label_style:Nnn` is used by the `label` key to creates the variables containing the *label style* and will allow to use `\arabic*`, `\Alph*`, `\alph*`, `\Roman*` and `\roman*` as arguments.

`__enumext_label_style:cvn`

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.

```

471 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
472 {
473   \tl_clear_new:N #1
474   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
475   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
476   \tl_map_inline:Nn \g__enumext_counter_styles_tl
477   {
478     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
479     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
480     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
481   }
482   \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
483   { \tl_use:N \g__enumext_widest_label_tl }
484   \tl_set_eq:cN { the #2 } #1
485 }
486 \cs_generate_variant:Nn \__enumext_label_style:Nnn { cvn }

```

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

12.12 Setting keys associated with label

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

```

labelsep
labelwidth
wrap-label
wrap-label*
487 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
488 {
489   \keys_define:nn { enumext / #1 }
490   {
491     font .tl_set:c = { l__enumext_label_font_style_#2_tl },
492     font .value_required:n = true,
493     labelsep .dim_set:c = { l__enumext_labelsep_#2_dim },
494     labelsep .initial:n = {0.3333em},
495     labelsep .value_required:n = true,
496     labelwidth .dim_set:c = { l__enumext_labelwidth_#2_dim },
497     labelwidth .value_required:n = true,
498     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
499     wrap-label .initial:n = {##1},
500     wrap-label .value_required:n = true,
501     wrap-label* .code:n = {
502       \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
503       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
504     },
505     wrap-label* .value_required:n = true,
506   }
507 }
508 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

- In this point, the following are set `__enumext_wrapper_label_X:n` which will be used by `__enumext_make_label:` for the different levels of the `enumext` environment and is set to `__enumext_wrapper_label_v:n` which will be used by `__enumext_keyans_make_label:` for `keyans` and `keyanspic` environments.

`align` The `align` key is implemented differently for “starred” and “non starred” environments.

```

509 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
510 {
511   \keys_define:nn { enumext / #1 }
512   {
513     align .choice:,
514     align / left .code:n =
515       {
516         \tl_clear:c { l__enumext_label_fill_left_#2_tl }
517         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
518       },
519     align / right .code:n =
520       {
521         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
522         \tl_clear:c { l__enumext_label_fill_right_#2_tl }
523       },
524     align / center .code:n =
525       {
526         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
527         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
528       },
529     align / unknown .code:n =
530       \msg_error:nneee { enumext } { unknown-choice }
531       { align } { left, ~ right, ~ center } { \exp_not:n {##1} },
532     align .initial:n = left,
533     align .value_required:n = true,
534   }
535 }
536 \clist_map_inline:nn
537 {
538   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
539 }
540 { \__enumext_tmp:nn #1 }

541 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
542 {
543   \keys_define:nn { enumext / #1 }
544   {
545     align .choice:,
```

```

546 align / left .code:n = \str_set:cn { l__enumext_align_label#2_str } { l },
547 align / right .code:n = \str_set:cn { l__enumext_align_label#2_str } { r },
548 align / center .code:n = \str_set:cn { l__enumext_align_label#2_str } { c },
549 align / unknown .code:n =
550     \msg_error:nneee { enumext } { unknown-choice }
551     { align } { left, ~ right, ~ center } { \exp_not:n {##1} },
552 align .initial:n = left,
553 align .value_required:n = true,
554 }
555 }
556 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

(End of definition for align.)

12.13 Setting label and ref keys

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

12.13.1 Define and set label and ref keys for enumext environment

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

```

label \l__enumext_label_i_tl 557 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
ref   \l__enumext_label_ii_tl 558 {
\l__enumext_label_iii_tl 559     \keys_define:nn { enumext / #1 }
\l__enumext_label_iv_tl 560     {
561         label .code:n = {
562             \__enumext_label_style:cnv { l__enumext_label#2_tl }
563             { l__enumext_counter#2_tl } {##1}
564             \dim_set_eq:cN { l__enumext_labelwidth#2_dim }
565             \l__enumext_current_widest_dim
566         },
567         label .initial:n = #3,
568         label .value_required:n = true,
569         ref .code:n = \__enumext_standar_ref:n {##1},
570         ref .value_required:n = true,
571     }
572 }
573 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
574 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*. ) }
575 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
576 \__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }

```

(End of definition for label and others.)

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

```

577 \cs_new_protected:Npn \__enumext_standar_ref:n #1
578 {
579     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
580     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
581     {
582         \msg_error:nnn { enumext } { key-ref-empty } { enumext }
583     }
584     {
585         \tl_set_eq:Nc
586             \l__enumext_ref_the_count_tl { l__enumext_counter_ \__enumext_level: _tl }
587         \__enumext_regex_counter_style:
588         \tl_set_eq:Nc
589             \l__enumext_ref_the_count_tl { l__enumext_the_counter_ \__enumext_level: _tl }
590         \tl_put_right:ce { l__enumext_renew_the_count_ \__enumext_level: _tl }
591         {
592             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
593             { \exp_not:V \l__enumext_ref_key_arg_tl }
594         }
595     }
596 }

```


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

```

597 \cs_new_protected:Nn \__enumext_standar_ref:
598 {
599   \tl_if_empty:cF { \__enumext_renew_the_count_ \__enumext_level: _tl }
600   {
601     \tl_use:c { \__enumext_renew_the_count_ \__enumext_level: _tl }
602   }
603 }

```

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

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

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

```

\l__enumext_label_vii_tl
\l__enumext_label_viii_tl
604 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
605 {
606   \keys_define:nn { enumext / #1 }
607   {
608     label .code:n = {
609       \__enumext_label_style:cvn { \__enumext_label_#2_tl }
610       { \__enumext_counter_#2_tl } {##1}
611       \dim_set_eq:cN { \__enumext_labelwidth_#2_dim }
612       \__enumext_current_widest_dim
613     },
614     label .initial:n = #3,
615     label .value_required:n = true,
616     ref .code:n = \__enumext_starred_ref:n {##1},
617     ref .value_required:n = true,
618   }
619 }
620 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
621 \__enumext_tmp:nnn { keyans* } { viii } { \Alph*.}

```

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

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

```

622 \cs_new_protected:Npn \__enumext_starred_ref:n #1
623 {
624   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
625   \int_compare:nNt { \l__enumext_level_h_int } = { 1 }
626   {
627     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
628     {
629       \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
630     }
631     {
632       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
633       \__enumext_regex_counter_style:
634       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
635       \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
636       {
637         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
638         { \exp_not:V \l__enumext_ref_key_arg_tl }
639       }
640     }
641   }
642   \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
643   {
644     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
645     {
646       \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
647     }
648     {
649       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
650       \__enumext_regex_counter_style:
651       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
652       \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
653       {
654         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
655         { \exp_not:V \l__enumext_ref_key_arg_tl }

```

```

656         }
657     }
658 }
659 }

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

660 \cs_new_protected:Nn \__enumext_starred_ref:
661 {
662     \int_compare:nNt { \__enumext_level_h_int } = { 1 }
663     {
664         \tl_if_empty:NF \__enumext_renew_the_count_vii_tl
665         {
666             \tl_use:N \__enumext_renew_the_count_vii_tl
667         }
668     }
669     \int_compare:nNt { \__enumext_keyans_level_h_int } = { 1 }
670     {
671         \tl_if_empty:NF \__enumext_renew_the_count_viii_tl
672         {
673             \tl_use:N \__enumext_renew_the_count_viii_tl
674         }
675     }
676 }

```

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

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

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

```

\l__enumext_label_v_tl
\l__enumext_label_vi_tl

677 \keys_define:nn { enumext / keyans }
678 {
679     label .code:n = {
680         \__enumext_label_style:cvn { \l__enumext_label_v_tl }
681         { \l__enumext_counter_v_tl } {#1}
682         \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
683         \l__enumext_current_widest_dim
684         \__enumext_label_style:cvn { \l__enumext_label_vi_tl }
685         { \l__enumext_counter_vi_tl } {#1}
686         \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
687         \l__enumext_current_widest_dim
688     },
689     label .initial:n = \Alph*,
690     label .value_required:n = true,
691     ref .code:n = \__enumext_keyans_ref:n {#1},
692     ref .value_required:n = true,
693 }

```

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

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

```

\__enumext_keyans_ref:n
\__enumext_keyans_ref:n

694 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
695 {
696     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
697     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
698     {
699         \msg_error:nnn { enumext } { key-ref-empty } { keyans }
700     }
701     {
702         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
703         \__enumext_regex_counter_style:
704         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
705         \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
706         {
707             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
708             { \exp_not:V \l__enumext_ref_key_arg_tl }
709         }
710     }
711 }

```

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

```

712 \cs_new_protected:Nn \__enumext_keyans_ref:
713 {
714   \tl_if_empty:NF \__enumext_renew_the_count_v_tl
715   {
716     \tl_use:N \__enumext_renew_the_count_v_tl
717   }
718 }

```

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

12.14 Setting start, start* and widest keys

```

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

```

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

```

#1: \__enumext_label_X_tl
#2: \__enumext_start_X_int
#3: <integer or string>

```

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

```

719 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
720 {
721   \__enumext_if_is_int:nTF { #3 }
722   {
723     \int_set:Nn #2 {#3}
724   }
725   {
726     \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
727     { \int_set:Nn #2 { \int_from_alph:n {#3} } }
728     \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
729     { \int_set:Nn #2 { \int_from_roman:n {#3} } }
730   }
731 }
732 \cs_generate_variant:Nn \__enumext_start_from:NNn { ccn, cce }

```

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

```

\__enumext_widest_from:nNNn
\__enumext_widest_from:nccn

```

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

```

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

```

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

```

733 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
734 {
735   \__enumext_if_is_int:nTF {#4}
736   {
737     \setcounter{enumX#1} { #4 }
738   }
739   {
740     \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
741     { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
742     \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
743     { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
744   }
745   \__enumext_label_width_by_box:cv
746   { \__enumext_labelwidth_#1_dim } { \__enumext_label_#1_tl }
747 }
748 \cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }

```

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

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

```

widest
start*
start
749 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
750 {

```

```

751 \keys_define:nn { enumext / #1 }
752 {
753   start* .code:n = {
754     \__enumext_start_from:ccn
755     { l__enumext_label_#2_tl }
756     { l__enumext_start_#2_int } {##1}
757   },
758   start* .value_required:n = true,
759   start .code:n = {
760     \__enumext_start_from:cce
761     { l__enumext_label_#2_tl }
762     { l__enumext_start_#2_int } { \int_eval:n {##1} }
763   },
764   start .initial:n = 1,
765   start .value_required:n = true,
766   widest .code:n = {
767     \__enumext_widest_from:nccn {#2}
768     { l__enumext_label_#2_tl }
769     { l__enumext_labelwidth_#2_dim } {##1}
770   },
771   widest .value_required:n = true,
772 }
773 }
774 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

12.15 Setting keys for vertical spaces

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

```

topsep 775 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
partopsep 776 {
parsep 777   \keys_define:nn { enumext / #1 }
noitemsep 778   {
nosep 779     topsep .skip_set:c = { l__enumext_topsep_#2_skip },
780     topsep .initial:n = {#3},
781     topsep .value_required:n = true,
782     partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
783     partopsep .initial:n = {#4},
784     partopsep .value_required:n = true,
785     parsep .skip_set:c = { l__enumext_parsep_#2_skip },
786     parsep .initial:n = {#5},
787     parsep .value_required:n = true,
788     itemsep .skip_set:c = { l__enumext_itemsep_#2_skip },
789     itemsep .initial:n = {#6},
790     itemsep .value_required:n = true,
791     noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
792     noitemsep .value_forbidden:n = true,
793     nosep .meta:n = {
794       itemsep = 0pt, parsep = 0pt,
795       topsep = 0pt, partopsep = 0pt,
796     },
797     nosep .value_forbidden:n = true,
798   }
799 }

```

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

```

800 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
801 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
802 { 4.0pt plus 2.0pt minus 1.0pt }
803 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
804 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
805 { 2.0pt plus 1.0pt minus 1.0pt }
806 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
807 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
808 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
809 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
810 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
811 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
812 { 2.0pt plus 1.0pt minus 1.0pt }
813 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }

```

```

814 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
815 { 4.0pt plus 2.0pt minus 1.0pt }
816 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
817 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
818 { 2.0pt plus 1.0pt minus 1.0pt }

```

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

12.16 Setting base-fix key

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

```

base-fix
\__enumext_nested_base_line_fix:
819 \cs_set_protected:Npn \__enumext_tmp:n #1
820 {
821   \keys_define:nn { enumext / #1 }
822   {
823     base-fix .bool_set:N = \__enumext_base_line_fix_bool,
824     base-fix .initial:n = false,
825     base-fix .value_forbidden:n = true,
826   }
827 }
828 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

```

829 \cs_new_protected:Nn \__enumext_nested_base_line_fix:
830 {
831   \bool_lazy_and:nnT
832   { \bool_if_p:N \__enumext_standar_first_bool }
833   { \bool_if_p:N \__enumext_base_line_fix_bool }
834   {
835     \mode_leave_vertical:
836     \vspace { -\baselineskip }
837     \keys_set:nn { enumext / level-1 }
838     {
839       topsep = 0pt, above = 0pt, above* = 0pt,
840     }
841   }
842   \bool_lazy_and:nnT
843   { \bool_if_p:N \__enumext_starred_first_bool }
844   { \bool_if_p:N \__enumext_base_line_fix_bool }
845   {
846     \mode_leave_vertical:
847     \vspace { -\baselineskip }
848     \keys_set:nn { enumext / enumext* }
849     {
850       topsep = 0pt, above = 0pt, above* = 0pt,
851     }
852   }
853   \bool_set_false:N \__enumext_base_line_fix_bool
854 }

```

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

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

12.17 Setting keys for horizontal spaces

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

```

855 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
856 {
857   \keys_define:nn { enumext / #1 }
858   {
859     itemindent .dim_set:c = { \__enumext_fake_item_indent_#2_dim },
860     itemindent .value_required:n = true,
861     rightmargin .dim_set:c = { \__enumext_rightmargin_#2_dim },

```

```

862     rightmargin .value_required:n = true,
863     listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
864     listparindent .value_required:n = true,
865     list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
866     list-offset .value_required:n = true,
867     list-indent .code:n =
868         \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
869         \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
870     list-indent .value_required:n = true,
871 }
872 }
873 \clist_map_inline:nn
874 {
875     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
876 }
877 { l__enumext_tmp:nn #1 }

```

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

For `enumext*` and `keyans*` environments the situation is a bit different, the `list-indent` key behaves like the `list-offset` key.

```

878 \cs_set_protected:Npn l__enumext_tmp:nn #1 #2
879 {
880     \keys_define:nn { enumext / #1 }
881     {
882         itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
883         itemindent .value_required:n = true,
884         rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
885         rightmargin .value_required:n = true,
886         listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
887         listparindent .value_required:n = true,
888         list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
889         list-offset .value_required:n = true,
890         list-indent .meta:n = { list-offset = ##1 },
891         list-indent .value_required:n = true,
892     }
893 }
894 \clist_map_inline:nn
895 {
896     {enumext*}{vii}, {keyans*}{viii}
897 }
898 { l__enumext_tmp:nn #1 }

```

12.17.1 Functions for setting the fake `itemindent`

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

```

899 \cs_set_protected:Nn l__enumext_fake_item:
900 {
901     \dim_compare:nNnT
902     { \dim_use:c { l__enumext_fake_item_indent_ l__enumext_level: _dim } }
903     >
904     { \c_zero_dim }
905     {
906         \tl_set:ce { l__enumext_fake_item_indent_ l__enumext_level: _tl }
907         {
908             \exp_not:N \mode_leave_vertical:
909             \exp_not:n { \skip_horizontal:n
910                 { \dim_use:c { l__enumext_fake_item_indent_ l__enumext_level: _dim } }
911             }
912             \ignorespaces
913         }
914     }
915 }
916 \cs_set_protected:Nn l__enumext_keyans_fake_item:
917 {
918     \dim_compare:nNnT
919     { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
920     {
921         \tl_set:Nc l__enumext_fake_item_indent_v_tl
922         {

```



```

922         \exp_not:N \mode_leave_vertical:
923         \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
924     }
925 }
926 }
927 \cs_set_protected:Nn \__enumext_fake_item_vii:
928 {
929     \dim_compare:nNnT
930     { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
931     {
932         \tl_set:Nc \l__enumext_fake_item_indent_vii_tl
933         {
934             \exp_not:N \mode_leave_vertical:
935             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
936         }
937     }
938 }
939 \cs_set_protected:Nn \__enumext_fake_item_viii:
940 {
941     \dim_compare:nNnT
942     { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
943     {
944         \tl_set:Nc \l__enumext_fake_item_indent_viii_tl
945         {
946             \exp_not:N \mode_leave_vertical:
947             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
948         }
949     }
950 }

```

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

12.18 Setting show-length key

show-length

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

```

951 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
952 {
953     \keys_define:nn { enumext / #1 }
954     {
955         show-length .bool_set:c = { \l__enumext_show_length_#2_bool },
956         show-length .initial:n = false,
957     }
958 }
959 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

12.19 Setting before, after and first keys

before

before*

after

first

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

```

960 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
961 {
962     \keys_define:nn { enumext / #1 }
963     {
964         before .tl_set:c = { \l__enumext_before_no_starred_key_#2_tl },
965         before .value_required:n = true,
966         before* .tl_set:c = { \l__enumext_before_starred_key_#2_tl },
967         before* .value_required:n = true,
968         after .tl_set:c = { \l__enumext_after_stop_list_#2_tl },
969         after .value_required:n = true,
970         first .tl_set:c = { \l__enumext_after_list_args_#2_tl },
971         first .value_required:n = true,
972     }
973 }
974 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

12.19.1 Functions for before, after and first keys in enumext

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

```

975 \cs_new_protected:Nn \__enumext_before_args_exec:
976 {
977   \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
978 }

```

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

```

979 \cs_new_protected:Nn \__enumext_before_keys_exec:
980 {
981   \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
982 }

```

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

```

983 \cs_new_protected:Nn \__enumext_after_stop_list:
984 {
985   \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
986 }

```

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

```

987 \cs_new_protected:Nn \__enumext_after_args_exec:
988 {
989   \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
990 }

```

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

12.19.2 Functions for before, after and first keys in keyans

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

```

\__enumext_before_args_exec_v:
\__enumext_before_keys_exec_v:
\__enumext_after_stop_list_v:
\__enumext_after_args_exec_v:
991 \cs_new_protected:Nn \__enumext_before_args_exec_v:
992 {
993   \tl_use:N \l__enumext_before_starred_key_v_tl
994 }
995 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
996 {
997   \tl_use:N \l__enumext_before_no_starred_key_v_tl
998 }
999 \cs_new_protected:Nn \__enumext_after_stop_list_v:
1000 {
1001   \tl_use:N \l__enumext_after_stop_list_v_tl
1002 }
1003 \cs_new_protected:Nn \__enumext_after_args_exec_v:
1004 {
1005   \tl_use:N \l__enumext_after_list_args_v_tl
1006 }

```

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

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

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

```

\__enumext_before_args_exec_vii:
\__enumext_before_keys_exec_vii:
\__enumext_after_stop_list_vii:
\__enumext_after_args_exec_vii:
1007 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
1008 {
1009   \tl_use:N \l__enumext_before_starred_key_vii_tl
1010 }
1011 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
1012 {
1013   \tl_use:N \l__enumext_before_starred_key_viii_tl
1014 }
1015 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
1016 {
1017   \tl_use:N \l__enumext_before_no_starred_key_vii_tl
1018 }
1019 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
1020 {

```

```

1021     \tl_use:N \l__enumext_before_no_starred_key_viii_tl
1022   }
1023   \cs_new_protected:Nn \__enumext_after_stop_list_vii:
1024   {
1025     \tl_use:N \l__enumext_after_stop_list_vii_tl
1026   }
1027   \cs_new_protected:Nn \__enumext_after_stop_list_viii:
1028   {
1029     \tl_use:N \l__enumext_after_stop_list_viii_tl
1030   }
1031   \cs_new_protected:Nn \__enumext_after_args_exec_vii:
1032   {
1033     \tl_use:N \l__enumext_after_list_args_vii_tl
1034   }
1035   \cs_new_protected:Nn \__enumext_after_args_exec_viii:
1036   {
1037     \tl_use:N \l__enumext_after_list_args_viii_tl
1038   }

```

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

12.20 Setting keys for multicols and minipage

The default value of the `columns-sep` key is handled by the state of the boolean variable `\l__enumext_columns_sep_X_bool` which is handled in the internal definition of the `enumext` and `keyans` environments. Define and set `mini-env`, `mini-sep`, `columns-sep` and `columns` keys for `enumext`, `enumext*`, `keyans` and `keyans*` environments.

```

1039   \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1040   {
1041     \keys_define:nn { enumext / #1 }
1042     {
1043       mini-env .dim_set:c = { l__enumext_minipage_right_#2_dim },
1044       mini-env .value_required:n = true,
1045       mini-sep .dim_set:c = { l__enumext_minipage_hsep_#2_dim },
1046       mini-sep .initial:n = 0.3333em,
1047       mini-sep .value_required:n = true,
1048       columns-sep .dim_set:c = { l__enumext_columns_sep_#2_dim },
1049       columns-sep .value_required:n = true,
1050       columns .int_set:c = { l__enumext_columns_#2_int },
1051       columns .initial:n = 1,
1052       columns .value_required:n = true,
1053     }
1054   }
1055   \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

For `enumext*` and `keyans*` environments the situation is a bit different, the command `\miniright` is not available, so we will add the keys `mini-right` and `mini-right*` to implement support for `minipage` environment.

```

1056   \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1057   {
1058     \keys_define:nn { enumext / #1 }
1059     {
1060       mini-right .tl_gset:c = { g__enumext_miniright_code_#2_tl },
1061       mini-right .value_required:n = true,
1062       mini-right* .code:n = {
1063         \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
1064         \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1065       },
1066       mini-right* .value_required:n = true,
1067     }
1068   }
1069   \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

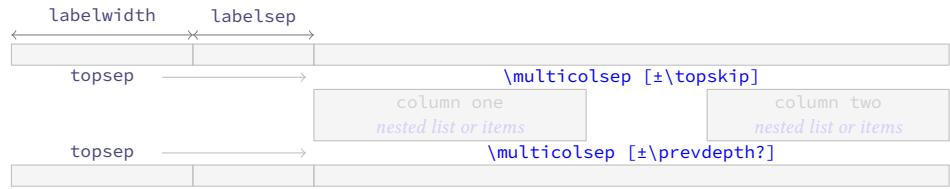
```

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

12.21 Adjustment of vertical spaces for multicols

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

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

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

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

12.21.1 Adjustment of vertical spaces for multicols in enumext

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

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

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

```

1070 \cs_new_protected:Nn \_enumext_multi_set_vskip:
1071 {
1072   \skip_set:cn { \l\_enumext_multicols_above_ \_enumext_level: } _skip {
1073     {
1074       \skip_use:c { \l\_enumext_topsep_ \_enumext_level: } _skip {
1075         }
1076       \skip_set:cn { \l\_enumext_multicols_below_ \_enumext_level: } _skip {
1077         {
1078           \skip_use:c { \l\_enumext_topsep_ \_enumext_level: } _skip {
1079             }
1080           \_enumext_add_pre_parsep:
1081         }
1082       }

```

(End of definition for `_enumext_multi_set_vskip:`)

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

```

1082 \cs_new_protected:Nn \_enumext_add_pre_parsep:
1083 {
1084   \int_case:nn { \l\_enumext_level_int }
1085   {
1086     { 2 }{
1087       \skip_if_eq:nnF { \l\_enumext_parsep_i_skip } { \c_zero_skip } {
1088         {
1089           \skip_add:Nn \l\_enumext_multicols_above_ii_skip { \l\_enumext_parsep_i_skip }
1090         }
1091       }
1092     { 3 }{
1093       \skip_if_eq:nnF { \l\_enumext_parsep_ii_skip } { \c_zero_skip } {
1094         {
1095           \skip_add:Nn \l\_enumext_multicols_above_iii_skip { \l\_enumext_parsep_ii_skip }
1096         }
1097       }
1098     { 4 }{
1099       \skip_if_eq:nnF { \l\_enumext_parsep_iii_skip } { \c_zero_skip } {
1100         {
1101           \skip_add:Nn \l\_enumext_multicols_above_iv_skip { \l\_enumext_parsep_iii_skip }
1102         }
1103       }
1104     }
1105   }

```

(End of definition for `_enumext_add_pre_parsep:`)

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

```

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

```

(End of definition for `__enumext_multi_addvspace:`.)

12.21.2 Adjustment of vertical spaces for multicols in keyans

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

`__enumext_keyans_multi_addvspace:`

```

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

```

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

12.22 Adjustment of vertical spaces for minipage

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

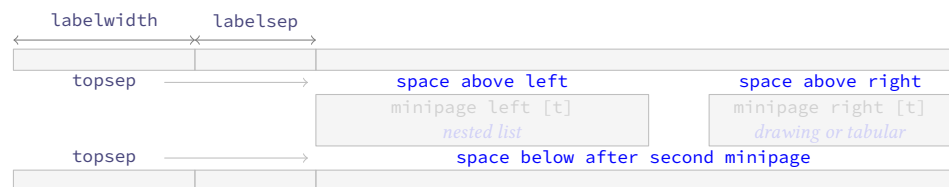


Figure 8: Representation of the `minipage` spacing adjustment for a nested level.

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

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

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

12.22.1 Adjustment of vertical spaces for minipage in enumext

`__enumext_minipage_set_skip:`
`__enumext_minipage_add_space:`

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

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

```

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

```

Now we will see if the environment `multicols` is active, if so we set `\topskip=0pt` and then we make `\multicolsep` have the same value as `\l__enumext_minipage_right_skip`

```

1165   \int_compare:nNtT
1166   { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1167   {
1168     \skip_zero:N \topskip
1169     \skip_set_eq:NN \multicolsep \l__enumext_minipage_right_skip
1170   }
1171 }

```

The function `__enumext_minipage_add_space:` will apply the spaces on the “left side” using `\addvspace` “above” the `__enumext_mini_env*` environment, taking into account whether \TeX is in $\langle horizontal mode \rangle$ or $\langle vertical mode \rangle$. 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*.

```

1172 \cs_new_protected:Nn \__enumext_minipage_add_space:
1173 {
1174   \__enumext_minipage_set_skip:
1175   \mode_if_vertical:TF
1176   {
1177     \nopagebreak\nointerlineskip
1178   }
1179   {
1180     \par\nopagebreak\nointerlineskip
1181     \skip_zero:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1182   }
1183   \addvspace{ 0.245\box_ht:N \strutbox }
1184 }

```

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

`__enumext_zero_parsep:`

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

```

1185 \cs_new_protected:Nn \__enumext_zero_parsep:

```



```

1186 {
1187   \int_case:nn { \l__enumext_level_int }
1188   {
1189     { 2 }{
1190       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1191       {
1192         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1193       }
1194     }
1195     { 3 }{
1196       \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1197       {
1198         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1199       }
1200     }
1201     { 4 }{
1202       \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1203       {
1204         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1205       }
1206     }
1207   }
1208 }

```

(End of definition for `__enumext_zero_parsep:`.)

12.22.2 Adjustment of vertical spaces for minipage in keyans

`__enumext_keyans_mini_set_vskip:`

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

```

1209 \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1210 {
1211   \skip_zero_new:N \l__enumext_minipage_after_skip
1212   \skip_zero_new:N \l__enumext_minipage_left_skip
1213   \skip_zero_new:N \l__enumext_minipage_right_skip
1214   \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1215   {
1216     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1217     {
1218       \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1219       \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1220       \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1221       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1222       {
1223         \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1224       }
1225     }
1226     {
1227       \skip_set:Nn \l__enumext_minipage_left_skip
1228       {
1229         \skip_use:N \l__enumext_topsep_v_skip
1230       }
1231       \skip_set:Nn \l__enumext_minipage_right_skip
1232       {
1233         0.705\box_dp:N \strutbox
1234       }
1235       \skip_set:Nn \l__enumext_minipage_after_skip
1236       {
1237         1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1238       }
1239     }
1240   }
1241   {
1242     \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1243     {
1244       \skip_set:Nn \l__enumext_minipage_left_skip
1245       {
1246         0.5\box_dp:N \strutbox
1247         + \l__enumext_partopsep_v_skip
1248       }
1249       \skip_set:Nn \l__enumext_minipage_right_skip

```

```

1250         {
1251             \l__enumext_partopsep_v_skip
1252         }
1253         \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1254     }
1255     {
1256         \skip_set:Nn \l__enumext_minipage_left_skip
1257         {
1258             0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1259         }
1260         \skip_set:Nn \l__enumext_minipage_right_skip
1261         {
1262             \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1263         }
1264         \skip_set:Nn \l__enumext_minipage_after_skip
1265         {
1266             0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1267         }
1268     }
1269 }
1270 }

```

(End of definition for `__enumext_keyans_mini_set_vskip:`)

`__enumext_keyans_mini_addvspace:`

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

```

1271 \cs_new_protected:Nn \__enumext_keyans_mini_addvspace:
1272 {
1273     \__enumext_keyans_mini_set_vskip:
1274     \mode_if_vertical:T
1275     {
1276         \skip_add:Nn \l__enumext_minipage_left_skip
1277         {
1278             \l__enumext_partopsep_v_skip
1279         }
1280         \skip_add:Nn \l__enumext_minipage_after_skip
1281         {
1282             \l__enumext_partopsep_v_skip
1283         }
1284     }
1285     \par\nopagebreak
1286     \addvspace { \l__enumext_minipage_left_skip }
1287 }

```

(End of definition for `__enumext_keyans_mini_addvspace:`)

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

`__enumext_mini_set_vskip_vii:`

`__enumext_mini_set_vskip_viii:`

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

```

1288 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1289 {
1290     \skip_zero_new:N \l__enumext_minipage_left_skip
1291     \skip_gzero_new:N \g__enumext_minipage_right_skip
1292     \skip_gzero_new:N \g__enumext_minipage_after_skip
1293     \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1294     {
1295         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1296         \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1297     }
1298     {
1299         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1300         \skip_gset:Nn \g__enumext_minipage_right_skip
1301         {
1302             \l__enumext_topsep_vii_skip
1303         }
1304         \skip_gset:Nn \g__enumext_minipage_after_skip

```

```

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

```

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

`__enumext_mini_addvspace_vii:`
`__enumext_mini_addvspace_viii:`

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

Here we will NOT take into account whether \TeX is in $\langle\textit{horizontal mode}\rangle$ or $\langle\textit{vertical mode}\rangle$, since `\partopsep` is equal to `0pt` in both environments.

```

1345 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1346 {
1347     \__enumext_mini_set_vskip_vii:
1348     \par\nopagebreak
1349     \addvspace { \l__enumext_minipage_left_skip }
1350 }
1351 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1352 {
1353     \__enumext_mini_set_vskip_viii:
1354     \par\nopagebreak
1355     \addvspace { \l__enumext_minipage_left_skip }
1356 }

```

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

12.22.4 The command `\miniright`

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

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

```

1357 \NewDocumentCommand \miniright { s }
1358 {
1359   \int_compare:nNt { \__enumext_keyans_pic_level_int } = { 1 }
1360   {
1361     \msg_error:nnn { enumext } { wrong-miniright-place }
1362   }
1363   % outside
1364   \bool_lazy_and:nnT
1365   { \int_compare_p:nNn { \__enumext_level_int } = { 0 } }
1366   { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
1367   {
1368     \msg_error:nnn { enumext } { wrong-miniright-place }
1369   }
1370   % starred env
1371   \bool_if:NT \__enumext_starred_bool
1372   {
1373     \msg_error:nnn { enumext } { wrong-miniright-starred }
1374   }
1375   \int_compare:nNtF { \__enumext_keyans_level_int } = { 1 }
1376   {
1377     \__enumext_keyans_mini_right_cmd:n {#1}
1378   }
1379   { \__enumext_mini_right_cmd:n {#1} }
1380 }

```

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

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

```

1381 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1382 {
1383   \dim_compare:nNtF
1384   { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1385   {
1386     \__enumext_multicols_stop:
1387     \end{__enumext_mini_env*}
1388     \hfill
1389     \begin{__enumext_mini_env*}
1390     { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } }
1391     % Add vertical space above
1392     \par\addvspace { \__enumext_minipage_right_skip }
1393     \bool_if:nF {#1}
1394     {
1395       \centering
1396     }
1397     \int_gzero:N \g__enumext_minipage_stat_int
1398   }
1399   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1400   % paranoia
1401   \RenewDocumentCommand \miniright { s }
1402   {
1403     \msg_error:nn { enumext } { many-miniright-used }
1404   }
1405 }

```

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

`__enumext_keyans_mini_right_cmd:n` The function `__enumext_keyans_mini_right_cmd:n` takes as argument the *starred* ‘*’ of the `\miniright` command in the `keyans` environment. The implementation of this function is the same as that of the `__enumext_mini_right_cmd:n` function of the `enumext` environment.

```

1406 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1407 {
1408   \dim_compare:nNtF { \__enumext_minipage_right_v_dim } > { \c_zero_dim }

```

```

1409     {
1410       \__enumext_keyans_multicols_stop:
1411       \end{\__enumext_mini_env*}
1412       \hfill
1413       \begin{\__enumext_mini_env*}{ \__enumext_minipage_right_v_dim }
1414       \par\addvspace { \__enumext_minipage_right_skip }
1415       \bool_if:nF {#1}
1416       {
1417         \centering
1418       }
1419       \int_gzero:N \g__enumext_minipage_stat_int
1420     }
1421     { \msg_error:nnn { enumext } { wrong-miniright-use } }
1422   \RenewDocumentCommand \miniright { s }
1423   {
1424     \msg_error:nn { enumext } { many-miniright-used }
1425   }
1426 }

```

(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* 1427 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
below 1428 {
below* 1429   \keys_define:nn { enumext / #1 }
1430   {
1431     above .skip_set:c = { \__enumext_vspace_above_#2_skip },
1432     above .value_required:n = true,
1433     above* .code:n      = \bool_set_true:c { \__enumext_vspace_a_star_#2_bool }
1434                     \keys_set:nn { enumext / #1 } { above = {##1} },
1435     above* .value_required:n = true,
1436     below .skip_set:c = { \__enumext_vspace_below_#2_skip },
1437     below .value_required:n = true,
1438     below* .code:n      = \bool_set_true:c { \__enumext_vspace_b_star_#2_bool }
1439                     \keys_set:nn { enumext / #1 } { below = {##1} },
1440     below* .value_required:n = true,
1441   }
1442 }
1443 \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.

```

1444 \cs_new_protected:Nn \__enumext_vspace_above:
1445 {
1446   \skip_if_eq:nnF
1447   { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1448   {
1449     \bool_if:cTF { \__enumext_vspace_a_star_ \__enumext_level: _bool }
1450     {
1451       \vspace*{ \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1452     }
1453     {
1454       \vspace { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1455     }
1456   }
1457 }

```

(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.

```

1458 \cs_new_protected:Nn \__enumext_vspace_below:
1459 {
1460   \skip_if_eq:nnF
1461   { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1462   {
1463     \bool_if:cTF { \__enumext_vspace_b_star_ \__enumext_level: _bool }
1464     {
1465       \vspace*{ \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1466     }
1467     {
1468       \vspace { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1469     }
1470   }
1471 }

```

(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.

```

1472 \cs_new_protected:Nn \__enumext_vspace_above_v:
1473 {
1474   \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1475   {
1476     \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1477     {
1478       \vspace*{ \l__enumext_vspace_above_v_skip }
1479     }
1480     { \vspace { \l__enumext_vspace_above_v_skip } }
1481   }
1482 }

```

(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.

```

1483 \cs_new_protected:Nn \__enumext_vspace_below_v:
1484 {
1485   \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1486   {
1487     \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1488     {
1489       \vspace*{ \l__enumext_vspace_below_v_skip }
1490     }
1491     { \vspace { \l__enumext_vspace_below_v_skip } }
1492   }
1493 }

```

(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:`

```

1494 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1495 {
1496   \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1497   {
1498     \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1499     {
1500       \vspace*{ \l__enumext_vspace_above_vii_skip }
1501     }
1502     { \vspace { \l__enumext_vspace_above_vii_skip } }
1503   }
1504 }
1505 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1506 {
1507   \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1508   {

```



```

1509         \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1510         {
1511             \vspace*{ \l__enumext_vspace_above_viii_skip }
1512         }
1513         { \vspace { \l__enumext_vspace_above_viii_skip } }
1514     }
1515 }

```

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

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

`__enumext_vspace_below_viii:`

```

1516 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1517 {
1518     \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1519     {
1520         \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1521         {
1522             \vspace*{ \l__enumext_vspace_below_vii_skip }
1523         }
1524         { \vspace { \l__enumext_vspace_below_vii_skip } }
1525     }
1526 }
1527 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1528 {
1529     \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1530     {
1531         \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1532         {
1533             \vspace*{ \l__enumext_vspace_below_viii_skip }
1534         }
1535         { \vspace { \l__enumext_vspace_below_viii_skip } }
1536     }
1537 }

```

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

12.24 Setting series, resume and resume* keys

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

`series`
`resume`
`resume*`

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

```

1538 \cs_set_protected:Npn \__enumext_tmp:n #1
1539 {
1540     \keys_define:nn { enumext / #1 }
1541     {
1542         series .str_set:N = \l__enumext_series_str,
1543         series .value_required:n = true,
1544         resume .code:n = \__enumext_resume_series:n {##1},
1545         resume* .code:n = \__enumext_resume_starred:,
1546         resume* .value_forbidden:n = true,
1547     }
1548 }
1549 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

12.24.1 Internal functions for series key

`__enumext_filter_series:n`
`__enumext_filter_series_key:n`
`__enumext_filter_series_pair:nn`

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.

```

1550 \cs_new:Npn \__enumext_filter_series:n #1
1551 {
1552     \use:e
1553     {
1554         \keyval_parse:NNn
1555         \__enumext_filter_series_key:n
1556         \__enumext_filter_series_pair:nn {#1}
1557     }
1558 }

```

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

```

1559 \cs_new:Npn \__enumext_filter_series_key:n #1
1560 {
1561   \str_case:nnF {#1}
1562   {
1563     { resume } {} { resume* } {} { base-fix } {}
1564   }
1565   { , { \exp_not:n {#1} } }
1566 }

```

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.

```

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

```

(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).

```

1576 \cs_new_protected:Npn \__enumext_parse_series:n #1
1577 {
1578   \str_if_empty:NTF \l__enumext_series_str
1579   {
1580     \bool_if:NF \l__enumext_resume_active_bool
1581     {
1582       \__enumext_resume_last:n {#1}
1583     }
1584   }
1585   {
1586     \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str_tl }
1587     \tl_gset:ce { g__enumext_series_ \l__enumext_series_str_tl }
1588     { \__enumext_filter_series:n {#1} }
1589     \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str_int }
1590     {
1591       \int_new:c { g__enumext_series_ \l__enumext_series_str_int }
1592     }
1593   }
1594 }

```

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.

```

1595 \cs_new_protected:Npn \__enumext_resume_last:n #1
1596 {
1597   \bool_if:NT \l__enumext_standar_first_bool
1598   {
1599     \tl_gclear:N \g__enumext_standar_series_tl
1600     \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1601   }
1602   \bool_if:NT \l__enumext_starred_first_bool
1603   {
1604     \tl_gclear:N \g__enumext_starred_series_tl

```

```

1605         \tl_gset:Nc \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1606     }
1607 }

```

(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_⟨series name⟩_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).

```

1608 \cs_new_protected:Nn \__enumext_resume_save_counter:
1609 {
1610     \bool_if:NT \g__enumext_standar_bool
1611     {
1612         \tl_if_empty:NF \l__enumext_series_str
1613         {
1614             \int_gset_eq:cN
1615             { g__enumext_series_ \l__enumext_series_str_int } \value{enumXi}
1616         }
1617         \tl_if_empty:NTF \l__enumext_resume_name_tl
1618         {
1619             \str_if_empty:NT \l__enumext_series_str
1620             {
1621                 \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1622             }
1623         }
1624         {
1625             \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl_int }
1626             {
1627                 \int_gset_eq:cN
1628                 { g__enumext_series_ \l__enumext_resume_name_tl_int } \value{enumXi}
1629             }
1630         }
1631         \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl_int }
1632         {
1633             \int_gset_eq:cN
1634             { g__enumext_resume_ \l__enumext_store_name_tl_int } \value{enumXi}
1635         }
1636     }
1637     \bool_if:NT \g__enumext_starred_bool
1638     {
1639         \tl_if_empty:NF \l__enumext_series_str
1640         {
1641             \int_gset_eq:cN
1642             { g__enumext_series_ \l__enumext_series_str_int } \value{enumXvii}
1643         }
1644         \tl_if_empty:NTF \l__enumext_resume_name_tl
1645         {
1646             \str_if_empty:NT \l__enumext_series_str
1647             {
1648                 \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1649             }
1650         }
1651         {
1652             \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl_int }
1653             {
1654                 \int_gset_eq:cN
1655                 { g__enumext_series_ \l__enumext_resume_name_tl_int } \value{enumXvii}
1656             }
1657         }
1658         \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl_int }
1659         {
1660             \int_gset_eq:cN

```

```

1661         { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1662     }
1663 }
1664 }

```

(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.

```

1665 \cs_new_protected:Npn \__enumext_resume_series:n #1
1666 {
1667     \tl_if_empty:NTF {#1}
1668     {
1669         \__enumext_resume_counter:n { }
1670     }
1671     {
1672         \tl_if_exist:CTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1673         {
1674             \__enumext_resume_counter:n {#1}
1675             \bool_if:NT \g__enumext_standar_bool
1676             {
1677                 \keys_set:nv { enumext / level-1 }
1678                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1679             }
1680             \bool_if:NT \g__enumext_starred_bool
1681             {
1682                 \keys_set:nv { enumext / enumext* }
1683                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1684             }
1685         }
1686         {
1687             \bool_if:NT \g__enumext_standar_bool
1688             {
1689                 \msg_error:nnn { enumext } { unknown-series } {#1}
1690             }
1691             \bool_if:NT \g__enumext_starred_bool
1692             {
1693                 \msg_error:nnn { enumext } { unknown-series } {#1}
1694             }
1695         }
1696     }
1697 }

```

(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`.

```

1698 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1699 {
1700     \bool_set_true:N \l__enumext_resume_active_bool
1701     \tl_set:Nn \l__enumext_resume_name_tl {#1}
1702     \tl_if_empty:NTF \l__enumext_resume_name_tl
1703     {
1704         \__enumext_resume_counter:
1705     }
1706     {
1707         \__enumext_resume_counter_series:
1708     }
1709     \__enumext_resume_counter_save_ans:
1710 }

```

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.

```

1711 \cs_new_protected:Nn \__enumext_resume_counter:
1712 {
1713   \bool_if:NT \g__enumext_standar_bool
1714   {
1715     \int_gincr:N \g__enumext_resume_int
1716     \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1717   }
1718   \bool_if:NT \g__enumext_starred_bool
1719   {
1720     \int_gincr:N \g__enumext_resume_vii_int
1721     \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1722   }
1723 }

```

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.

```

1724 \cs_new_protected:Nn \__enumext_resume_counter_series:
1725 {
1726   \bool_if:NT \g__enumext_standar_bool
1727   {
1728     \int_set:Nn \l__enumext_start_i_int
1729     {
1730       \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1731     }
1732   }
1733   \bool_if:NT \g__enumext_starred_bool
1734   {
1735     \int_set:Nn \l__enumext_start_vii_int
1736     {
1737       \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1738     }
1739   }
1740 }

```

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.

```

1741 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1742 {
1743   \bool_lazy_and:nnT
1744   { \bool_if_p:N \l__enumext_standar_first_bool }
1745   { \bool_if_p:N \l__enumext_store_active_bool }
1746   {
1747     \int_set:Nn \l__enumext_start_i_int
1748     {
1749       \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1750     }
1751   }
1752   \bool_lazy_and:nnT
1753   { \bool_if_p:N \l__enumext_starred_first_bool }
1754   { \bool_if_p:N \l__enumext_store_active_bool }
1755   {
1756     \int_set:Nn \l__enumext_start_vii_int
1757     {
1758       \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1759     }
1760   }
1761 }

```

(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.

```

1762 \cs_new_protected:Nn \__enumext_resume_starred:
1763 {

```

```

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

```

(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*`.

```

1781 \cs_set_protected:Npn \__enumext_tmp:n #1
1782 {
1783     \keys_define:nn { enumext / #1 }
1784     {
1785         save-ans .code:n = \__enumext_storing_set:n {##1},
1786         save-ans .value_required:n = true,
1787     }
1788 }
1789 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

12.25.2 Internal functions for save-ans key

__enumext_start_save_ans_msg:
__enumext_stop_save_ans_msg:

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

```

1790 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
1791 {
1792     \msg_term:nnVV { enumext } { save-ans-log }
1793     \g__enumext_envir_name_tl \l__enumext_store_name_tl
1794 }
1795 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
1796 {
1797     \msg_term:nnVV { enumext } { save-ans-log-hook }
1798     \g__enumext_envir_name_tl \g__enumext_store_name_tl
1799 }

```

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

__enumext_storing_set:n
__enumext_storing_exec:

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

```

1800 \cs_new_protected:Npn \__enumext_storing_set:n #1
1801 {
1802     \tl_set:Nx \l__enumext_store_name_tl {#1}
1803     \tl_if_empty:NTF \l__enumext_store_name_tl
1804     {
1805         \bool_lazy_or:nnT
1806         { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1807         {

```

```

1808         \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
1809     }
1810 }
1811 {
1812     \bool_lazy_or:nnT
1813     { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1814     {
1815         \__enumext_start_save_ans_msg:
1816         \__enumext_storing_exec:
1817     }
1818 }
1819 }

```

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*`.

```

1820 \cs_new_protected:Nn \__enumext_storing_exec:
1821 {
1822     \bool_set_true:N \l__enumext_store_active_bool
1823     \bool_set_true:N \l__enumext_check_answers_bool
1824     \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1825     \__enumext_anskey_env_make:V \l__enumext_store_name_tl
1826     \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1827     {
1828         \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
1829         \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1830     }
1831     \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1832     {
1833         \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
1834         \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1835     }
1836     \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1837     {
1838         \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
1839         \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1840     }
1841 }

```

(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

no-store

```

1842 \cs_set_protected:Npn \__enumext_tmp:n #1
1843 {
1844   \keys_define:nn { enumext / #1 }
1845   {
1846     check-ans .bool_set:N = \l__enumext_check_ans_key_bool,
1847     check-ans .initial:n = false,
1848     check-ans .value_required:n = true,
1849     no-store .code:n = {
1850       \bool_set_false:N \l__enumext_check_answers_bool
1851       \bool_set_false:N \l__enumext_check_ans_key_bool
1852     },
1853     no-store .value_forbidden:n = true,
1854   }
1855 }
1856 \clist_map_inline:nn
1857 {
1858   level-1, level-2, level-3, level-4, enumext*
1859 }
1860 { \__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 `\l__enumext_store_name_tl`, that is, the `save-ans` key is active, if so it will check the state of the variable `\l__enumext_check_answers_bool` handled by the key `no-store` and will execute the function `__enumext_check_ans_level:` only if “true”, i.e. the key `no-store` is not active.

```

1861 \cs_new_protected:Nn \__enumext_check_ans_active:
1862 {
1863   \tl_if_empty:NF \l__enumext_store_name_tl
1864   {
1865     \bool_if:NT \l__enumext_check_answers_bool
1866     {
1867       \__enumext_check_ans_level:
1868     }
1869   }
1870 }
```

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

```

1871 \cs_new_protected:Nn \__enumext_check_ans_level:
1872 {
1873   \int_case:nn { \l__enumext_level_int }
1874   {
1875     { 1 } {
1876       \bool_lazy_all:nT
1877       {
1878         { \bool_if_p:N \g__enumext_starred_bool }
1879         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
1880       }
1881       {
1882         \int_gdecr:N \g__enumext_item_number_int
1883         \bool_set_false:N \l__enumext_item_number_bool
1884       }
1885     }
1886     { 2 } {
1887       \int_gdecr:N \g__enumext_item_number_int
1888       \bool_set_false:N \l__enumext_item_number_bool
1889     }
1890     { 3 } {
```

```

1891         \int_gdecr:N \g__enumext_item_number_int
1892         \bool_set_false:N \l__enumext_item_number_bool
1893     }
1894     { 4 }{
1895         \int_gdecr:N \g__enumext_item_number_int
1896         \bool_set_false:N \l__enumext_item_number_bool
1897     }
1898 }

```

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.

```

1899 \int_case:nn { \l__enumext_level_h_int }
1900 {
1901     { 1 }{
1902         \bool_lazy_all:nT
1903         {
1904             { \bool_if_p:N \g__enumext_standar_bool }
1905             { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
1906         }
1907         {
1908             \int_gdecr:N \g__enumext_item_number_int
1909             \bool_set_false:N \l__enumext_item_number_bool
1910         }
1911     }
1912 }
1913 }

```

(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.

```

1914 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
1915 {
1916     \bool_lazy_and:nnT
1917     { \bool_if_p:N \l__enumext_check_ans_key_bool }
1918     { \bool_if_p:N \g__enumext_standar_bool }
1919     {
1920         \bool_gset_true:N \g__enumext_check_ans_key_bool
1921     }
1922     \bool_lazy_and:nnT
1923     { \bool_if_p:N \l__enumext_check_ans_key_bool }
1924     { \bool_if_p:N \g__enumext_starred_bool }
1925     {
1926         \bool_gset_true:N \g__enumext_check_ans_key_bool
1927     }
1928 }

```

(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:`.

```

1929 \cs_new_protected:Nn \__enumext_item_answer_diff:
1930 {
1931     \int_gset:Nn \g__enumext_item_answer_diff_int
1932     {
1933         \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
1934     }
1935 }

```

(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:`.

```

1936 \cs_new_protected:Nn \__enumext_check_ans_show:
1937 {
1938     \int_case:nn { \g__enumext_item_answer_diff_int }

```

```

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

```

(End of definition for __enumext_check_ans_show: and others.)

__enumext_check_ans_log: The function __enumext_check_ans_log: will be executed within the function __enumext_execute_after_env: when the key `check-ans` is not active, that is, when `\g__enumext_check_ans_key_bool` is “false” and write in the log the appropriate message according to the value of `\g__enumext_item_answer_diff_int` set by the function `__enumext_item_answer_diff:`.

```

1960 \cs_new_protected:Nn \__enumext_check_ans_log:
1961 {
1962     \int_case:nn { \g__enumext_item_answer_diff_int }
1963     {
1964         { -1 }{ \__enumext_check_ans_log_msg_less: }
1965         { 0 }{ \__enumext_check_ans_log_msg_same_ok: }
1966         { 1 }{ \__enumext_check_ans_log_msg_greater: }
1967     }
1968 }
1969 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
1970 {
1971     \msg_log:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
1972     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1973 }
1974 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
1975 {
1976     \msg_log:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
1977     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1978 }
1979 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
1980 {
1981     \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
1982     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1983 }

```

(End of definition for __enumext_check_ans_log: and others.)

12.25.6 Check for \item* and \anspic* commands

__enumext_check_starred_cmd:n The function __enumext_check_starred_cmd:n performs an extra check for the `keyans`, `keyans*` and `keyanspic` environments. Unlike the check executed by `check-ans` key this one is not controlled by any key, it is intended to prevent the forgetting of `\item*` or `\anspic*` in these environments.

```

1984 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
1985 {
1986     \int_compare:nNnT
1987     { \g__enumext_check_starred_cmd_int } = { 0 }
1988     {
1989         \msg_warning:nnnV
1990         { enumext } { missing-starred }{ #1 } { \l__enumext_check_start_line_env_tl }
1991     }
1992     \int_compare:nNnT
1993     { \g__enumext_check_starred_cmd_int } > { 1 }
1994     {
1995         \msg_warning:nnnV

```

```

1996         { enumext } { many-starred }{ #1 } \l__enumext_check_start_line_env_tl
1997     }
1998     \int_gzero:N \g__enumext_check_starred_cmd_int
1999     \tl_clear:N \l__enumext_check_start_line_env_tl
2000 }

```

(End of definition for \l__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*`.

```

2001 \cs_set_protected:Npn \l__enumext_tmp:n #1
2002 {
2003     \keys_define:nn { enumext / #1 }
2004     {
2005         wrap-ans .cs_set_protected:Np = \l__enumext_anskey_wrapper:n ##1,
2006         wrap-ans .initial:n =
2007             {
2008                 \fbox{\parbox[t]{\dimeval{\itemwidth -2\fboxsep -2\fboxrule}}{##1}}
2009             },
2010         wrap-ans .value_required:n = true,
2011         wrap-opt .cs_set_protected:Np = \l__enumext_keyans_wrapper_opt:n ##1,
2012         wrap-opt .initial:n = [{##1}],
2013         wrap-opt .value_required:n = true,
2014         save-sep .tl_set:N = \l__enumext_store_keyans_item_opt_sep_tl,
2015         save-sep .initial:n = {, ~ },
2016         save-sep .value_required:n = true,
2017         mark-ans .tl_set:N = \l__enumext_mark_answer_sym_tl,
2018         mark-ans .initial:n = \textasteriskcentered,
2019         mark-ans .value_required:n = true,
2020         mark-pos .choice:,
2021         mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2022         mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2023         mark-pos / unknown .code:n =
2024             \msg_error:nneee { enumext } { unknown-choice }
2025             { mark-pos } { left, ~ right } { \exp_not:n {##1} },
2026         mark-pos .initial:n = right,
2027         mark-pos .value_required:n = true,
2028         show-ans .bool_set:N = \l__enumext_show_answer_bool,
2029         show-ans .initial:n = false,
2030         show-ans .value_required:n = true,
2031         show-pos .bool_set:N = \l__enumext_show_position_bool,
2032         show-pos .initial:n = false,
2033         show-pos .value_required:n = true,
2034         mark-ref .tl_set:N = \l__enumext_mark_ref_sym_tl,
2035         mark-ref .initial:n = \textasteriskcentered,
2036         mark-ref .value_required:n = true,
2037         save-ref .bool_set:N = \l__enumext_store_ref_key_bool,
2038         save-ref .initial:n = false,
2039         save-ref .value_required:n = true,
2040     }
2041 }
2042 \clist_map_inline:nn { level-1, enumext* } { \l__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`.

```

2043 \cs_set_protected:Npn \l__enumext_tmp:n #1
2044 {
2045     \keys_define:nn { enumext / #1 }
2046     {
2047         mark-pos .choice:,
2048         mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2049         mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2050         mark-pos .initial:n = right,
2051         mark-pos .value_required:n = true,
2052         show-ans .bool_set:N = \l__enumext_show_answer_bool,
2053         show-ans .initial:n = false,
2054         show-ans .value_required:n = true,
2055         show-pos .bool_set:N = \l__enumext_show_position_bool,

```

```

2056         show-pos .initial:n = false,
2057         show-pos .value_required:n = true,
2058     }
2059 }
2060 \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*.

```

2061 \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2062 {
2063     \bool_if:cF { \__enumext_store_save_key_ \__enumext_level: _bool }
2064     {
2065         \tl_clear:c { \__enumext_save_key_ \__enumext_level: _tl }
2066         \tl_set:ce
2067             { \__enumext_store_save_key_ \__enumext_level: _tl }
2068             { \__enumext_filter_save_key:n {#1} }
2069     }
2070 }
2071 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2072 {
2073     \bool_if:NF \__enumext_store_save_key_vii_bool
2074     {
2075         \tl_clear:N \__enumext_store_save_key_vii_tl
2076         \tl_set:Ne \__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2077     }
2078 }

```

(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.

```

2079 \cs_set_protected:Npn \__enumext_tmp:n #1
2080 {
2081     \keys_define:nn { enumext / enumext* }
2082     {
2083         save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2084         save-key .value_required:n = true,
2085     }
2086     \keys_define:nn { enumext / #1 }
2087     {
2088         save-key .code:n = \__enumext_parse_save_key:n {##1},
2089         save-key .value_required:n = true,
2090     }
2091 }
2092 \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*`.

```

2093 \cs_new_protected:Npn \__enumext_parse_save_key:n #1

```

```

2094 {
2095   \bool_set_true:c { \__enumext_store_save_key_ \__enumext_level: _bool }
2096   \tl_clear:c { \__enumext_save_key_ \__enumext_level: _tl }
2097   \tl_set:ce
2098     { \__enumext_store_save_key_ \__enumext_level: _tl }
2099     { \__enumext_filter_save_key:n {#1} }
2100 }
2101 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2102 {
2103   \bool_set_true:N \__enumext_store_save_key_vii_bool
2104   \tl_clear:N \__enumext_store_save_key_vii_tl
2105   \tl_set:Nx \__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2106 }

```

(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
2107 \cs_new:Npn \__enumext_filter_save_key:n #1
2108 {
2109   \use:e
2110   {
2111     \keyval_parse:NNn
2112       \__enumext_filter_save_key_key:n
2113       \__enumext_filter_save_key_pair:nn {#1}
2114   }
2115 }

```

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.

```

2116 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2117 {
2118   \str_case:nnF {#1}
2119   {
2120     { resume } {} { resume* } {} { no-store } {} { base-fix } {}
2121   }
2122   { , { \exp_not:n {#1} } }
2123 }

```

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.

```

2124 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2125 {
2126   \str_case:nnF {#1}
2127   {
2128     { series } {} { resume } {} { save-ans } {} { save-ref } {}
2129     { save-key } {} { check-ans } {} { show-ans } {} { show-pos } {}
2130     { wrap-ans } {} { mark-ans } {} { wrap-opt } {} { save-sep } {}
2131     { mark-ref } {} { mini-env } {} { mini-sep } {} { mini-right } {}
2132     { mini-right* } {}
2133   }
2134   { , { \exp_not:n {#1} } } = { \exp_not:n {#2} } }
2135 }

```

(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

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

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

```

2136 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2137 {
2138   \prop_gput_if_not_in:cen { g__enumext_ \__enumext_store_name_tl _prop }
2139   {
2140     \int_eval:n { \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop } + 1 }

```

```

2141     }
2142     { #1 }
2143 }
2144 \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.

```

2145 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2146 {
2147   \seq_gput_right:cn { g__enumext_ \__enumext_store_name_tl_seq } { #1 }
2148 }
2149 \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.

```

2150 \cs_new_protected:Npn \__enumext_store_level_open:
2151 {
2152   \bool_if:NT \l__enumext_check_answers_bool
2153   {
2154     \tl_if_empty:cTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2155     {
2156       \__enumext_store_addto_seq:n
2157       {
2158         \item \begin{enumext}
2159       }
2160     }
2161     {
2162       \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2163       {
2164         \item \begin{enumext} [
2165       }
2166       \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2167       {
2168         ]
2169       }
2170       \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2171     }
2172   }
2173 }
2174 \cs_new_protected:Npn \__enumext_store_level_close:
2175 {
2176   \bool_if:NT \l__enumext_check_answers_bool
2177   {
2178     \__enumext_store_addto_seq:n { \end{enumext} }
2179   }
2180 }

```

(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.

```

2181 \cs_new_protected:Npn \__enumext_store_level_open_vii:
2182 {
2183   \bool_if:NT \l__enumext_check_answers_bool
2184   {
2185     \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2186     {
2187       \__enumext_store_addto_seq:n
2188       {
2189         \item \begin{enumext*}
2190       }

```



```

2191     }
2192     {
2193         \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2194         {
2195             \item \begin{enumext*}[
2196             ]
2197             \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2198             {
2199                 ]
2200             }
2201             \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2202         }
2203     }
2204 }
2205 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2206 {
2207     \bool_if:NT \l__enumext_check_answers_bool
2208     {
2209         \__enumext_store_addto_seq:n { \end{enumext*} }
2210     }
2211 }

```

(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`

```

2212 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2213 {
2214     \mode_leave_vertical:
2215     \skip_horizontal:n { -\dim_use:N #2 }
2216     \makebox[0pt][ r ]
2217     {
2218         \makebox[ \dim_use:N #1 ] [ \l__enumext_mark_position_str ]
2219         {
2220             \tl_use:N \l__enumext_mark_answer_sym_tl
2221         }
2222     }
2223     \skip_horizontal:n { \dim_use:N #2 }
2224 }
2225 \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`.

```

2226 \cs_new_protected:Nn \__enumext_store_internal_ref:
2227 {
2228     \cs_set_protected:Npn \__enumext_tmp:n ##1
2229     {
2230         \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2231         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2232         \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2233         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2234     }
2235     \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2236     \cs_set:Npn \__enumext_tmp:n ##1
2237     { . \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.

```

2238     \bool_lazy_all:nT
2239     {

```

```

2240     { \bool_if_p:N \g__enumext_starred_bool }
2241     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2242   }
2243   {
2244     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2245     { \tl_use:N \l__enumext_label_copy_vii_tl }
2246   }
2247   \bool_lazy_all:nT
2248   {
2249     { \bool_not_p:n { \g__enumext_standar_bool } }
2250     { \bool_if_p:N \l__enumext_standar_bool }
2251     { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2252   }
2253   {
2254     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2255     {
2256       \tl_use:N \l__enumext_label_copy_vii_tl
2257       \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2258     }
2259   }

```

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.

```

2260   \bool_lazy_all:nT
2261   {
2262     { \bool_if_p:N \g__enumext_standar_bool }
2263     { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2264     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2265   }
2266   {
2267     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2268     {
2269       \tl_use:N \l__enumext_label_copy_i_tl
2270       \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2271     }
2272   }
2273   \cs_set:Npn \__enumext_tmp:n ##1
2274   { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } . }
2275   \bool_lazy_all:nT
2276   {
2277     { \bool_if_p:N \g__enumext_standar_bool }
2278     { \bool_if_p:N \l__enumext_starred_bool }
2279     { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2280   }
2281   {
2282     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2283     {
2284       \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2285       \tl_use:N \l__enumext_label_copy_vii_tl
2286     }
2287   }

```

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

```

2288   \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2289   {
2290     \l__enumext_store_name_tl \c_colon_str
2291     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2292   }

```

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.

```

2293   \tl_put_right:Ne \l__enumext_write_aux_file_tl
2294   {
2295     \__enumext_newlabel:nn
2296     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2297     { \l__enumext_newlabel_arg_two_tl }
2298   }
2299   \l__enumext_write_aux_file_tl
2300 }

```

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

```

2301 \cs_new_protected:Npn \__enumext_store_anskey_code:n #1
2302 {
2303   \int_gincr:N \g__enumext_item_anskey_int
2304   \__enumext_store_addto_prop:n {#1}
2305   \bool_if:NT \l__enumext_store_ref_key_bool
2306   {
2307     \__enumext_store_internal_ref:
2308   }
2309   \__enumext_anskey_show_wrap_left:n { #1 }

```

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

```

2310   \tl_clear:N \l__enumext_store_anskey_arg_tl
2311   \bool_lazy_and:nnT
2312   { \bool_if_p:N \l__enumext_store_columns_break_bool }
2313   { \bool_not_p:n { \l__enumext_starred_bool } }
2314   {
2315     \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2316   }
2317   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { \item }

```

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

```

2318   \bool_lazy_and:nnT
2319   { \bool_not_p:n { \l__enumext_starred_bool } }
2320   { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2321   {
2322     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2323     {
2324       ( \exp_not:V \l__enumext_store_item_join_int )
2325     }
2326   }

```

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

```

2327   \bool_if:NTF \l__enumext_store_item_star_bool
2328   {
2329     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2330     \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2331     {
2332       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2333       {
2334         [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2335       }
2336     }
2337     \dim_compare:nT
2338     {
2339       \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2340     }
2341     {
2342       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2343       {
2344         [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2345       }
2346     }
2347     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2348   }
2349   {
2350     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2351   }

```

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

```

2352   \bool_lazy_and:nnT

```

```

2353 { \bool_if_p:N \l__enumext_store_ref_key_bool }
2354 { \bool_if_p:N \l__enumext_hyperref_bool }
2355 {
2356   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2357   {
2358     \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2359     { \exp_not:V \l__enumext_mark_ref_sym_tl }
2360   }
2361 }
2362 \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2363 }

```

(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 *⟨argument⟩* passed to `\anskey` and the *⟨body⟩* for `anskey*` when using the `wrap-ans` key.

```

2364 \cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
2365 {
2366   \par
2367   \bool_if:NTF \l__enumext_starred_bool
2368   {
2369     \__enumext_print_keyans_box:NN \l__enumext_labelwidth_vii_dim \l__enumext_labelsep_vii_dim
2370   }
2371   {
2372     \__enumext_print_keyans_box:cc
2373     { \l__enumext_labelwidth_ \l__enumext_level: _dim }
2374     { \l__enumext_labelsep_ \l__enumext_level: _dim }
2375   }
2376   \__enumext_anskey_wrapper:n { #1 }
2377 }

```

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

```

2378 \cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
2379 {
2380   \bool_if:NT \l__enumext_show_answer_bool
2381   {
2382     \__enumext_anskey_show_wrap_arg:n { #1 }
2383   }
2384   \bool_if:NT \l__enumext_show_position_bool
2385   {
2386     \tl_set:Ne \l__enumext_mark_answer_sym_tl
2387     {
2388       \group_begin:
2389       \exp_not:N \normalfont
2390       \exp_not:N \footnotesize [ \int_eval:n
2391       {
2392         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2393       }
2394       ]
2395       \group_end:
2396     }
2397     \__enumext_anskey_show_wrap_arg:n { #1 }
2398   }
2399 }

```

(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 *⟨sequences⟩* and can (more or less) manage the options passed to each level, it is necessary that we have a little more control over `\item` when storing.

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

```

\__enumext_anskey_unknown:n
\__enumext_anskey_unknown:n

```

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

```

2400 \keys_define:nn { enumext / anskey }
2401 {
2402   break-col .bool_set:N = \__enumext_store_columns_break_bool,
2403   break-col .default:n = true,
2404   break-col .value_forbidden:n = true,
2405   item-join .int_set:N = \__enumext_store_item_join_int,
2406   item-join .value_required:n = true,
2407   item-star .bool_set:N = \__enumext_store_item_star_bool,
2408   item-star .default:n = true,
2409   item-star .value_forbidden:n = true,
2410   item-sym* .tl_set:N = \__enumext_store_item_symbol_tl,
2411   item-sym* .value_required:n = true,
2412   item-pos* .dim_set:N = \__enumext_store_item_symbol_sep_dim,
2413   item-pos* .value_required:n = true,
2414   unknown .code:n = { \__enumext_anskey_unknown:n {#1} },
2415 }

```

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

```

2416 \cs_new_protected:Npn \__enumext_anskey_unknown:n #1
2417 {
2418   \exp_args:NV \__enumext_anskey_unknown:nn \l_keys_key_str {#1}
2419 }
2420 \cs_new_protected:Npn \__enumext_anskey_unknown:nn #1 #2
2421 {
2422   \tl_if_blank:nTF {#2}
2423   {
2424     \msg_error:nnn { enumext } { anskey-cmd-key-unknown } {#1}
2425   }
2426   {
2427     \msg_error:nnnn { enumext } { anskey-cmd-key-value-unknown } {#1} {#2}
2428   }
2429 }

```

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

```

2430 \NewDocumentCommand \anskey { o +m }
2431 {
2432   \__enumext_anskey_safe_outer:
2433   \group_begin:
2434     \bool_if:NT \l__enumext_check_answers_bool
2435     {
2436       \tl_if_novalue:nF {#1}
2437       {
2438         \keys_set:nn { enumext / anskey } {#1}
2439       }
2440       \tl_if_blank:nTF {#2}
2441       {
2442         \msg_error:nn { enumext } { anskey-empty-arg }
2443       }
2444       {
2445         \__enumext_anskey_safe_inner:
2446         \__enumext_store_anskey_code:n {#2}
2447       }
2448     }
2449   \group_end:
2450 }

```

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

12.29.1 Internal functions for the command

`__enumext_anskey_safe_outer:`

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

`__enumext_anskey_safe_inner:`

```

2451 \cs_new_protected:Nn \__enumext_anskey_safe_outer:
2452 {
2453   \bool_if:NF \l__enumext_store_active_bool
2454   {
2455     \msg_error:nnnn { enumext } { anskey-wrong-place } { anskey } { enumext }
2456   }
2457   \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
2458   {
2459     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans }
2460   }
2461   \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
2462   {
2463     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans* }
2464   }
2465   \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
2466   {
2467     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyanspic }
2468   }
2469 }

```

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.

```

2470 \cs_new_protected:Nn \__enumext_anskey_safe_inner:
2471 {
2472   \int_incr:N \l__enumext_anskey_level_int
2473   \int_compare:nNt { \l__enumext_anskey_level_int } > { 1 }
2474   {
2475     \msg_error:nn { enumext } { anskey-nested }
2476   }
2477   \bool_if:NF \l__enumext_item_number_bool
2478   {
2479     \msg_error:nn { enumext } { anskey-unnumber-item }
2480   }
2481   \mode_if_math:T
2482   {
2483     \msg_error:nne { enumext } { anskey-math-mode } { \c_backslash_str anskey }
2484   }
2485 }

```

(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.

```

2486 \cs_new_protected:Nn \__enumext_undefine_anskey_env:
2487 {
2488   \cs_undefine:c { anskey* }
2489   \cs_undefine:c { endanskey* }
2490   \cs_undefine:c { __scontents_anskey*_env_begin: }
2491   \cs_undefine:c { __scontents_anskey*_env_end: }
2492 }

```

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

```

2493 \__enumext_before_env:nn { enumext }
2494 {
2495   \bool_lazy_and:nnT
2496   { \int_compare_p:nNn { \__enumext_level_int } = { 0 } }
2497   { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
2498   {
2499     \cs_if_free:cF { __scontents_anskey*_env_begin: }
2500     {
2501       \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2502     }
2503   }
2504 }
2505 \__enumext_before_env:nn { enumext* }
2506 {
2507   \bool_lazy_and:nnT
2508   { \int_compare_p:nNn { \__enumext_level_int } = { 0 } }
2509   { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
2510   {
2511     \cs_if_free:cF { __scontents_anskey*_env_begin: }
2512     {
2513       \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2514     }
2515   }
2516 }

```

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.

```

2517 \__enumext_before_env:nn { anskey* }
2518 {
2519   \int_compare:nNnT { \__enumext_keyans_level_int } = { 1 }
2520   {
2521     \msg_error:nnn { enumext } { anskey-env-wrong } { keyans }
2522   }
2523   \int_compare:nNnT { \__enumext_keyans_level_h_int } = { 1 }
2524   {
2525     \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
2526   }
2527   \int_compare:nNnT { \__enumext_keyans_pic_level_int } = { 1 }
2528   {
2529     \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
2530   }
2531   \bool_if:NF \__enumext_item_number_bool
2532   {
2533     \msg_error:nn { enumext } { anskey-unnumber-item }
2534   }
2535   \mode_if_math:T
2536   {
2537     \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
2538   }
2539 }

```

(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`.

```

2540 \cs_new_protected:Npn \__enumext_anskey_env_make:n #1
2541 {
2542   \bool_if:NT \__enumext_anskey_env_bool
2543   {
2544     \newenvsc{anskey*}[store-env=#1,print-env=false]
2545     \__enumext_anskey_env_exec:
2546   }
2547 }
2548 \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`.

```

2549 \cs_new_protected:Nn \__enumext_anskey_env_define_keys:
2550 {
2551   \keys_define:nn { scontents / scontents }
2552   {
2553     break-col .bool_gset:N = \g__enumext_store_columns_break_bool,
2554     break-col .default:n   = true,
2555     break-col .value_forbidden:n = true,
2556     item-join .int_gset:N = \g__enumext_store_item_join_int,
2557     item-join .value_required:n = true,
2558     item-star .bool_gset:N = \g__enumext_store_item_star_bool,
2559     item-star .default:n   = true,
2560     item-star .value_forbidden:n = true,
2561     item-sym* .tl_gset:N   = \g__enumext_store_item_symbol_tl,
2562     item-sym* .value_required:n = true,
2563     item-pos* .dim_gset:N = \g__enumext_store_item_symbol_sep_dim,
2564     item-pos* .value_required:n = true,
2565     print-env .undefine:,
2566     store-env .undefine:,
2567     write-out .undefine:,
2568     unknown   .code:n      = { \__enumext_anskey_env_unknown:n {##1} },
2569   }
2570 }

```

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`.

```

2571 \cs_new_protected:Npn \__enumext_anskey_env_unknown:n #1
2572 {
2573   \exp_args:NV \__enumext_anskey_env_unknown:nn \l_keys_key_str {#1}
2574 }
2575 \cs_new_protected:Npn \__enumext_anskey_env_unknown:nn #1#2
2576 {
2577   \tl_if_blank:nTF {#2}
2578   {
2579     \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
2580   }
2581   {
2582     \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
2583   }
2584 }

```

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`.

```

2585 \cs_new_protected:Nn \__enumext_anskey_env_reset_keys:
2586 {
2587   \keys_define:nn { scontents / scontents }
2588   {
2589     break-col .undefine:,
2590     item-join .undefine:,
2591     item-star .undefine:,
2592     item-sym* .undefine:,
2593     item-pos* .undefine:,
2594     write-out .code:n      = {
2595       \bool_set_false:N \l__scontents_storing_bool
2596       \bool_set_true:N  \l__scontents_writing_bool
2597       \tl_set:Nn \l__scontents_fname_out_tl {##1}
2598     },
2599     write-out .value_required:n = true,
2600     print-env .meta:nn         = { scontents } { print-env = ##1 },
2601     print-env .default:n       = true,
2602     store-env .meta:nn         = { scontents } { store-env = ##1 },
2603     unknown   .code:n          = { \__scontents_parse_environment_keys:n {##1} },
2604   }
2605 }

```

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*.

```

2606 \cs_new_protected:Npn \__enumext_rescan_anskey_env:n #1

```

```

2607 {
2608   \group_begin:
2609   \int_set:Nn \tex_newlinechar:D { `\\^^J }
2610   \__scontents_rescan_tokens:x
2611   {
2612     \endgroup % This assumes \catcode`\\=0... Things might go off otherwise.
2613     #1
2614   }
2615 }

```

(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)`.

```

2616 \cs_new_protected:Nn \__enumext_anskey_env_exec:
2617 {
2618   \__enumext_before_env:nn { anskey* }
2619   {
2620     \__enumext_anskey_env_define_keys:
2621   }

```

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.

```

2622   \hook_if_empty:nF {env/anskey*/after}
2623   {
2624     \hook_gremove_code:nn {env/anskey*/after} { * }
2625   }
2626   \__enumext_after_env:nn { anskey* }
2627   {
2628     \__enumext_anskey_env_save_keys:
2629     \tl_clear:N \l__enumext_store_anskey_env_tl
2630     \tl_clear:N \l__enumext_store_anskey_opt_tl
2631     \bool_if:NT \l__enumext_check_answers_bool
2632     {
2633       \tl_gset:Ne \l__enumext_store_anskey_env_tl
2634       {
2635         \seq_item:ce { g__scontents_name_ \l__enumext_store_name_tl _seq } { -1 }
2636       }
2637       \regex_match:nVTF
2638       { ^\s* \z | ^\s* \u{c__scontents_hidden_space_str} \z }
2639       \l__enumext_store_anskey_env_tl
2640       {
2641         \msg_error:nn { enumext } { anskey-empty-arg }
2642       }
2643       {
2644         \__enumext_anskey_env_store:
2645       }
2646     }
2647     \__enumext_anskey_env_clean_vars:
2648     \__enumext_anskey_env_reset_keys:
2649   }
2650 }

```

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`.

```

2651 \cs_new_protected:Nn \__enumext_anskey_env_save_keys:
2652 {
2653   \bool_lazy_and:nnT
2654   { \bool_if_p:N \g__enumext_store_columns_break_bool }
2655   { \bool_not_p:n { \l__enumext_starred_bool } }
2656   {
2657     \tl_put_left:Ne \l__enumext_store_anskey_opt_tl { ,break-col, }
2658   }

```

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

```

2659   \bool_lazy_and:nnT
2660   { \bool_not_p:n { \l__enumext_starred_bool } }
2661   { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }
2662   {
2663     \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2664     {
2665       ,item-join = \exp_not:V \g__enumext_store_item_join_int,
2666     }
2667   }

```

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

```

2668   \bool_if:NT \g__enumext_store_item_star_bool
2669   {
2670     \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2671     {
2672       ,item-star,
2673     }
2674     \tl_if_empty:NF \g__enumext_store_item_symbol_tl
2675     {
2676       \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2677       {
2678         ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,
2679       }
2680     }
2681     \dim_compare:nT
2682     {
2683       \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
2684     }
2685     {
2686       \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2687       {
2688         ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
2689       }
2690     }
2691   }
2692 }

```

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`.

```

2693 \cs_new_protected:Nn \__enumext_anskey_env_store:
2694 {
2695   \group_begin:
2696   \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
2697   {
2698     \exp_args:Ne
2699     \__enumext_store_anskey_code:n
2700     {
2701       \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2702     }
2703   }
2704   {
2705     \keys_set_known:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
2706     \exp_args:Ne
2707     \__enumext_store_anskey_code:n
2708     {
2709       \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2710     }
2711   }
2712   \group_end:
2713 }

```

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

```

2714 \cs_new_protected:Nn \__enumext_anskey_env_clean_vars:
2715 {
2716   \bool_gset_false:N \g__enumext_store_columns_break_bool
2717   \int_gzero:N       \g__enumext_store_item_join_int
2718   \bool_gset_false:N \g__enumext_store_item_star_bool
2719   \tl_gclear:N       \g__enumext_store_item_symbol_tl

```

```

2720     \dim_gzero:N          \g__enumext_store_item_symbol_sep_dim
2721 }

```

(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.

```

2722 \cs_new_protected:Nn \__enumext_execute_after_env:
2723 {
2724     \int_compare:nNnT { \__enumext_level_int } = { 0 }
2725     {
2726         \tl_if_empty:NF \g__enumext_store_name_tl
2727         {
2728             \__enumext_stop_save_ans_msg:
2729             \__enumext_item_answer_diff:
2730             \__enumext_log_global_vars:
2731             \__enumext_log_answer_vars:
2732             \bool_if:NTF \g__enumext_check_ans_key_bool
2733             {
2734                 \__enumext_check_ans_show:
2735             }
2736             { \__enumext_check_ans_log: }
2737             \__enumext_undefine_anskey_env:
2738         }
2739         \__enumext_reset_global_vars:
2740     }
2741 }

```

(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`.

```

2742 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2743 {
2744     \tl_clear:N \l__enumext_store_current_label_tl
2745     \int_compare:nNnTF { \__enumext_keyans_pic_level_int } = { 1 }
2746     {
2747         \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
2748     }
2749     {
2750         \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
2751     }
2752     \tl_if_novalue:nF { #1 }
2753     {
2754         % Set save-sep
2755         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2756         {
2757             \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
2758         }
2759         \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2760     }
2761     \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
2762 }

```

(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 $\langle label \rangle$. The mechanism defined here will allow to execute `\ref{\store name : position}` and will return 1. (A).

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 $\langle labels \rangle$ and remove the dots “.” from them, we do not want to get double dots in our references.

```

2763 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2764 {
2765   \bool_if:NT \l__enumext_store_ref_key_bool
2766   {
2767     \cs_set_protected:Npn \__enumext_tmp:n ##1
2768     {
2769       \tl_set_eq:cc { \__enumext_label_copy_##1_tl } { \__enumext_label_##1_tl }
2770       \tl_reverse:c { \__enumext_label_copy_##1_tl }
2771       \tl_remove_once:cn { \__enumext_label_copy_##1_tl } { . }
2772       \tl_reverse:c { \__enumext_label_copy_##1_tl }
2773     }
2774     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2775     \__enumext_keyans_store_ref_aux_i:
2776   }
2777 }

```

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

```

2778 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2779 {
2780   \bool_if:NT \g__enumext_starred_bool
2781   {
2782     \tl_set_eq:NN \__enumext_label_copy_i_tl \__enumext_label_copy_vii_tl
2783   }
2784   \int_compare:nNnT { \__enumext_keyans_pic_level_int } = { 1 }
2785   {
2786     \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2787     { \__enumext_label_copy_i_tl . \__enumext_label_copy_vi_tl }
2788   }
2789   \int_compare:nNnT { \__enumext_keyans_level_int } = { 1 }
2790   {
2791     \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2792     { \__enumext_label_copy_i_tl . \__enumext_label_copy_v_tl }
2793   }
2794   \int_compare:nNnT { \__enumext_keyans_level_h_int } = { 1 }
2795   {
2796     \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
2797     { \__enumext_label_copy_i_tl . \__enumext_label_copy_viii_tl }
2798   }
2799   \tl_put_right:Ne \__enumext_newlabel_arg_one_tl
2800   {
2801     \__enumext_store_name_tl \c_colon_str
2802     \int_eval:n { \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop } }
2803   }
2804   \__enumext_keyans_store_ref_aux_ii:
2805 }

```

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.

```

2806 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2807 {
2808   \tl_put_right:Ne \l__enumext_write_aux_file_tl
2809   {
2810     \__enumext_newlabel:nn
2811     { \exp_not:V \__enumext_newlabel_arg_one_tl }
2812     { \__enumext_newlabel_arg_two_tl }
2813   }
2814   \l__enumext_write_aux_file_tl
2815 }

```

(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 $\langle label \rangle$ `\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 $\langle contents \rangle$ 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.

```
2816 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2817 {
2818   \tl_clear:N \l__enumext_store_current_label_tl
2819   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2820   {
2821     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
2822   }
2823   {
2824     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
2825   }
2826   \tl_if_novalue:nF { #1 }
2827   {
2828     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2829     {
2830       \tl_put_right:Ne \l__enumext_store_current_label_tl
2831       {
2832         \l__enumext_store_keyans_item_opt_sep_tl
2833       }
2834     }
2835     \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2836   }
2837   \__enumext_keyans_addto_seq_link:
2838 }
```

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.

```
2839 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2840 {
2841   \bool_lazy_and:nnT
2842   { \bool_if_p:N \l__enumext_store_ref_key_bool }
2843   { \bool_if_p:N \l__enumext_hyperref_bool }
2844   {
2845     \tl_put_right:Ne \l__enumext_store_current_label_tl
2846     {
2847       \hfill \exp_not:N \hyperlink
2848       {
2849         \exp_not:V \l__enumext_newlabel_arg_one_tl
2850       }
2851       { \exp_not:V \l__enumext_mark_ref_sym_tl }
2852     }
2853   }
2854   \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
2855   \bool_if:NT \l__enumext_check_answers_bool
2856   {
2857     \int_gincr:N \g__enumext_item_anskey_int
2858   }
2859 }
```

(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 $\langle label \rangle$ 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 $\langle position \rangle$ of stored content in $\langle prop list \rangle$ for `keyans` and `keyanspic`. Need add `1` to `\g__enumext_⟨store name⟩_prop` for `show-pos` key.

```
2860 \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
2861 {
2862   \tl_if_novalue:nF { #1 }
2863   {
```

```

2864         \tl_set:Nc \l__enumext_store_current_opt_arg_tl { #1 }
2865     }
2866     \bool_if:NT \l__enumext_show_answer_bool
2867     {
2868         \__enumext_keyans_show_ans:
2869     }
2870     \bool_if:NT \l__enumext_show_position_bool
2871     {
2872         \__enumext_keyans_show_pos:
2873     }
2874 }
2875 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
2876 {
2877     \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
2878     {
2879         \bool_lazy_or:nnT
2880         { \bool_if_p:N \l__enumext_show_answer_bool }
2881         { \bool_if_p:N \l__enumext_show_position_bool }
2882         {
2883             \__enumext_keyans_wrapper_opt:n { \l__enumext_store_current_opt_arg_tl } \c_space_tl
2884         }
2885     }
2886 }
2887 \cs_new_protected:Nn \__enumext_keyans_show_ans:
2888 {
2889     \bool_if:NT \l__enumext_starred_bool
2890     {
2891         \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
2892         \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
2893     }
2894     \tl_put_left:Nn \l__enumext_label_v_tl
2895     {
2896         \__enumext_print_keyans_box:NN
2897         \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2898     }
2899 }
2900 \cs_new_protected:Nn \__enumext_keyans_show_pos:
2901 {
2902     \bool_if:NT \l__enumext_starred_bool
2903     {
2904         \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
2905         \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
2906     }
2907     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2908     {
2909         \tl_set:Nc \l__enumext_mark_answer_sym_tl
2910         {
2911             \group_begin:
2912             \exp_not:N \normalfont
2913             \exp_not:N \footnotesize [ \int_eval:n
2914                 {
2915                     \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2916                 }
2917             ]
2918             \group_end:
2919         }
2920     }
2921     {
2922         \tl_set:Nc \l__enumext_mark_answer_sym_tl
2923         {
2924             \group_begin:
2925             \exp_not:N \normalfont
2926             \exp_not:N \footnotesize [ \int_eval:n
2927                 {
2928                     \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
2929                 }
2930             ]
2931             \group_end:
2932         }
2933     }
2934     \tl_put_left:Nn \l__enumext_label_v_tl

```



```

2935     {
2936         \__enumext_print_keyans_box:NN
2937         \__enumext_labelwidth_i_dim \__enumext_labelsep_i_dim
2938     }
2939 }

```

(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`.

```

2940 \cs_new_protected:Npn \__enumext_default_item:n #1
2941 {
2942     \tl_if_novalue:nTF {#1}
2943     {
2944         \bool_if:NT \l__enumext_check_answers_bool
2945         {
2946             \int_gincr:N \g__enumext_item_number_int
2947             \bool_set_true:N \l__enumext_item_number_bool
2948         }
2949         \bool_set_true:c { \l__enumext_wrap_label_ \__enumext_level: _bool }
2950         \__enumext_item_std:w \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
2951     }
2952     {
2953         \bool_set_eq:cc
2954         { \l__enumext_wrap_label_ \__enumext_level: _bool }
2955         { \l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2956         \__enumext_item_std:w [#1] \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
2957     }
2958 }

```

(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`.

```

2959 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2960 {
2961     \tl_if_novalue:nTF {#1}
2962     {
2963         \tl_gset_eq:Nc
2964         \g__enumext_item_symbol_aux_tl { \l__enumext_item_symbol_ \__enumext_level: _tl }
2965     }
2966     {
2967         \tl_gset:Nn \g__enumext_item_symbol_aux_tl {#1}
2968     }
2969     \tl_if_novalue:nTF {#2}
2970     {
2971         \dim_set_eq:cc
2972         { \l__enumext_item_symbol_sep_ \__enumext_level: _dim }
2973         { \l__enumext_labelsep_ \__enumext_level: _dim }

```

```

2974     }
2975     {
2976         \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2977     }
2978     \bool_if:NT \__enumext_check_answers_bool
2979     {
2980         \int_gincr:N \g__enumext_item_number_int
2981         \bool_set_true:N \__enumext_item_number_bool
2982     }
2983     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2984     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2985 }

```

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

```

2986 \cs_new_protected:Nn \__enumext_item_star_exec:
2987 {
2988     \tl_if_empty:cF { l__enumext_item_symbol_ \__enumext_level: _tl }
2989     {
2990         \mode_leave_vertical:
2991         \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2992         \makebox[ 0pt ][ r ] { \g__enumext_item_symbol_aux_tl }
2993         \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
2994     }
2995 }

```

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

`__enumext_redefine_item:`
`__enumext_make_label`

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

```

2996 \cs_new_protected:Nn \__enumext_redefine_item:
2997 {
2998     \RenewDocumentCommand \item { s o o }
2999     {
3000         \bool_if:nTF {##1}
3001         {
3002             \__enumext_starred_item:nn {##2} {##3}
3003         }
3004         { \__enumext_default_item:n {##2} }
3005     }
3006 }

```

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

```

3007 \cs_new_protected:Nn \__enumext_make_label:
3008 {
3009     \RenewDocumentCommand \make_label { m }
3010     {
3011         \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
3012         \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3013         \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3014         {
3015             \__enumext_item_star_exec:
3016             \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3017         }
3018         { ##1 }
3019         \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
3020         \tl_gclear:N \g__enumext_item_symbol_aux_tl
3021     }
3022 }

```

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

🌱 This functions are passed to `__enumext_list_arg_two_X:` used in the definition of the `enumext` environment (§12.38).

12.34 Setting `item-sym*` and `item-pos*` keys

In order to have a cleaner implementation of `\item*` for the `enumext` and `enumext*` environments it is best to define a couple of keys that allow us to control and set by default the `<symbol>` and its `<offset>`.

`item-sym*` Define and set `item-sym*` and `item-pos*` keys for `enumext` and `enumext*`.

```

3023 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
3024 {

```

```

3025 \keys_define:nn { enumext / #1 }
3026 {
3027   item-sym* .tl_set:c = { l__enumext_item_symbol_#2_tl },
3028   item-sym* .value_required:n = true,
3029   item-sym* .initial:n = {$\star$},
3030   item-pos* .dim_set:c = { l__enumext_item_symbol_sep_#2_dim },
3031   item-pos* .value_required:n = true,
3032 }
3033 }
3034 \clist_map_inline:nn
3035 {
3036   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
3037 }
3038 { \__enumext_tmp:nn #1 }

```

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

12.35 Handling unknown keys

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

12.35.1 Handling unknown keys for keyans and keyans*

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

unknown
 __enumext_keyans_unknown_keys:n
 __enumext_keyans_unknown_keys:nn

```

3039 \cs_set_protected:Npn \__enumext_tmp:n #1
3040 {
3041   \keys_define:nn { enumext / #1 }
3042   {
3043     unknown .code:n = { \__enumext_keyans_unknown_keys:n {#1} }
3044   }
3045 }
3046 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }

```

Internal functions for handling unknown key.

```

3047 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:n #1
3048 {
3049   \exp_args:NV \__enumext_keyans_unknown_keys:nn \l_keys_key_str {#1}
3050 }
3051 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:nn #1#2
3052 {
3053   \tl_if_blank:nTF {#2}
3054   {
3055     \msg_error:nnn { enumext } { keyans-unknown-key } {#1}
3056   }
3057   {
3058     \msg_error:nnnn { enumext } { keyans-unknown-key-value } {#1} {#2}
3059   }
3060 }

```

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

12.35.2 Handling unknown keys for enumext*

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

unknown
 __enumext_starred_unknown_keys:n
 __enumext_starred_unknown_keys:nn

```

3061 \keys_define:nn { enumext / enumext* }
3062 {
3063   unknown .code:n = { \__enumext_starred_unknown_keys:n {#1} }
3064 }

```

Internal functions for handling unknown key.

```

3065 \cs_new_protected:Npn \__enumext_starred_unknown_keys:n #1
3066 {
3067   \exp_args:NV \__enumext_starred_unknown_keys:nn \l_keys_key_str {#1}
3068 }
3069 \cs_new_protected:Npn \__enumext_starred_unknown_keys:nn #1#2
3070 {
3071   \tl_if_blank:nTF {#2}
3072   {
3073     \msg_error:nnn { enumext } { starred-unknown-key } {#1}
3074   }
3075   {
3076     \msg_error:nnnn { enumext } { starred-unknown-key-value } {#1} {#2}
3077   }

```

```
3078 }
```

(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 3079 \cs_set_protected:Npn \__enumext_tmp:n #1
\__enumext_standar_unknown_keys:nn 3080 {
3081   \keys_define:nn { enumext / #1 }
3082   {
3083     unknown .code:n = { \__enumext_standar_unknown_keys:n {##1} }
3084   }
3085 }
3086 \clist_map_inline:nn { level-1,level-2,level-3,level-4 } { \__enumext_tmp:n {#1} }
```

Internal functions for handling `unknown` key.

```
3087 \cs_new_protected:Npn \__enumext_standar_unknown_keys:n #1
3088 {
3089   \exp_args:NV \__enumext_standar_unknown_keys:nn \l_keys_key_str {#1}
3090 }
3091 \cs_new_protected:Npn \__enumext_standar_unknown_keys:nn #1#2
3092 {
3093   \tl_if_blank:nTF {#2}
3094   {
3095     \msg_error:nnn { enumext } { standar-unknown-key } {#1}
3096   }
3097   {
3098     \msg_error:nnnn { enumext } { standar-unknown-key-value } {#1} {#2}
3099   }
3100 }
```

(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`.

```
3101 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
3102 {
3103   \tl_if_novalue:nTF { #1 }
3104   {
3105     \bool_set_true:N \__enumext_wrap_label_v_bool
3106     \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl
3107   }
3108   {
3109     \bool_set_eq:NN \__enumext_wrap_label_v_bool \__enumext_wrap_label_opt_v_bool
3110     \__enumext_item_std:w [#1] \tl_use:N \__enumext_fake_item_indent_v_tl
3111   }
3112 }
```

(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⟩`.

```
3113 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
3114 {
3115   \tl_set_eq:NN \__enumext_store_current_label_tmp_tl \__enumext_label_v_tl
3116   \__enumext_keyans_show_left:n { #1 }
3117   \bool_set_true:N \__enumext_wrap_label_v_bool
3118   \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl \__enumext_keyans_show_item
```

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

```
3119   \tl_set_eq:NN \__enumext_label_v_tl \__enumext_store_current_label_tmp_tl
3120   \__enumext_keyans_addto_prop:n { #1 }
3121   \__enumext_keyans_store_ref:
```

```

3122     \__enumext_keyans_addto_seq:n { #1 }
3123     \int_gincr:N \g__enumext_check_starred_cmd_int
3124 }

```

(End of definition for __enumext_keyans_starred_item:n.)

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

```

3125 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
3126 {
3127     \RenewDocumentCommand \item { s o }
3128     {
3129         \bool_if:nTF {##1}
3130         {
3131             \peek_remove_spaces:n
3132             {
3133                 \__enumext_keyans_starred_item:n {##2}
3134             }
3135         }
3136         {
3137             \__enumext_keyans_default_item:n {##2}
3138         }
3139     }
3140 }

```

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

```

3141 \cs_new_protected:Nn \__enumext_keyans_make_label:
3142 {
3143     \RenewDocumentCommand \makeLabel { m }
3144     {
3145         \tl_use:N \l__enumext_label_fill_left_v_tl
3146         \tl_use:N \l__enumext_label_font_style_v_tl
3147         \bool_if:NTF \l__enumext_wrap_label_v_bool
3148         {
3149             \__enumext_wrapper_label_v:n { ##1 }
3150         }
3151         { ##1 }
3152         \tl_use:N \l__enumext_label_fill_right_v_tl
3153     }
3154 }

```

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

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

12.37 Second argument of the lists

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

12.37.1 Calculation of \leftmargin and \itemindent

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

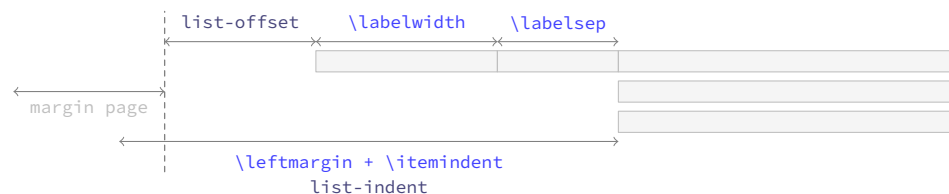
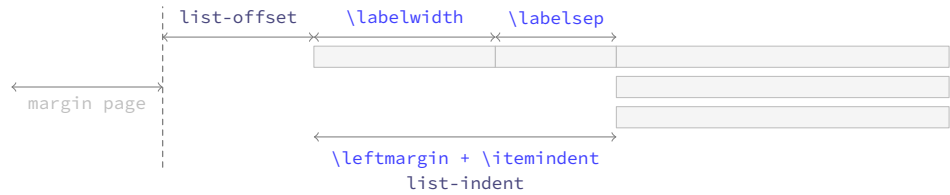
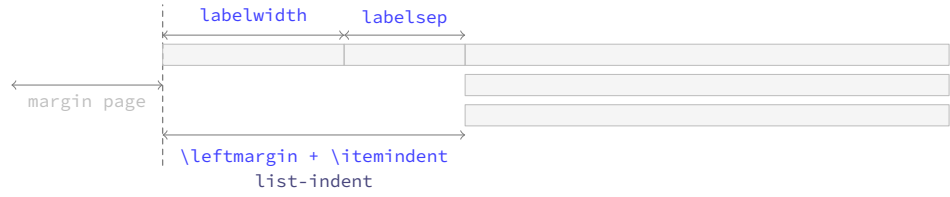


Figure 9: Representation of standard horizontal lengths in *list* environment.

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

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

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

```
\__enumext_calc_hspace:NNNNNNN
\__enumext_calc_hspace:ccccc
```

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

```
#1: \__enumext_labelwidth_X_dim      #2: \__enumext_labelsep_X_dim
#3: \__enumext_listoffset_X_dim      #4: \__enumext_leftmargin_tmp_X_dim
#5: \__enumext_leftmargin_X_dim      #6: \__enumext_itemindent_X_dim
#7: \__enumext_leftmargin_tmp_X_bool
```

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

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

```
3155 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
3156 {
3157   \dim_compare:nNt { #1 } < { \c_zero_dim }
3158   {
3159     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
3160     \dim_set:Nn #1 { \dim_abs:n { #1 } }
3161   }
3162   \dim_compare:nNt { #2 } < { \c_zero_dim }
3163   {
3164     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
3165     \dim_set:Nn #2 { \dim_abs:n { #2 } }
3166   }
3167 }
```

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

```
3167 \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }
```

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

```
3168 \dim_compare:nNtTF { #4 } < { \c_zero_dim }
3169 {
3170   \dim_set:Nn #6 { #1 + #2 - #4 }
3171   \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3172 }
3173 {
3174   \dim_compare:nNt { #4 } = { #1 + #2 }
3175   { \dim_set:Nn #6 { \c_zero_dim } }
3176   \dim_compare:nNt { #4 } < { #1 + #2 }
3177   { \dim_set:Nn #6 { #1 + #2 - #4 } }
3178   \dim_compare:nNt { #4 } > { #1 + #2 }
3179   {
3180     \dim_set:Nn #6 { -#1 - #2 + #4 }
3181     \dim_set:Nn #6 { #6*-1 }
3182   }
3183   \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3184 }
3185 }
3186 \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.

```

3187 \cs_set_protected:Npn \__enumext_tmp:n #1
3188 {
3189   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3190   {
3191     \__enumext_calc_hspace:ccccc
3192     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3193     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3194     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3195     { \__enumext_leftmargin_tmp_#1_bool }
3196     \clist_map_inline:nn
3197       { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3198       { \dim_set_eq:cc {###1} { \__enumext_###1_#1_dim } }
3199     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3200       { \skip_set_eq:cc {###1} { \__enumext_###1_#1_skip } }
3201     \usecounter { enumX#1 }
3202     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
3203     \str_if_eq:nnTF {#1} { v }
3204     {
3205       \__enumext_keyans_redefine_item:
3206       \__enumext_keyans_make_label:
3207       \__enumext_keyans_ref:
3208       \__enumext_keyans_fake_item:
3209       \bool_if:cT { \__enumext_show_length_#1_bool }
3210       {
3211         \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3212       }
3213     }
3214     {
3215       \__enumext_redefine_item:
3216       \__enumext_make_label:
3217       \__enumext_standar_ref:
3218       \__enumext_fake_item:
3219       \bool_if:cT { \__enumext_show_length_#1_bool }
3220       {
3221         \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \__enumext_level_int }
3222       }
3223     }
3224   }
3225 }
3226 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

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

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

```

3227 \cs_set_protected:Npn \__enumext_tmp:n #1
3228 {
3229   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3230   {
3231     \bool_set_true:c { \__enumext_leftmargin_tmp_#1_bool }
3232     \dim_zero:c { \__enumext_leftmargin_tmp_#1_dim }
3233     \__enumext_calc_hspace:ccccc
3234     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3235     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3236     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3237     { \__enumext_leftmargin_tmp_#1_bool }
3238     \clist_map_inline:nn
3239       { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3240       { \dim_set_eq:cc {###1} { \__enumext_###1_#1_dim } }
3241     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3242       { \skip_set_eq:cc {###1} { \__enumext_###1_#1_skip } }
3243     \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
3244     \skip_zero:N \partopsep
3245     \usecounter { enumX#1 }
3246     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }

```



```

3247     \__enumext_starred_ref:
3248     \str_if_eq:nnTF {#1} { vii }
3249     {
3250         \__enumext_fake_item_vii:
3251         \bool_if:cT { \__enumext_show_length_vii_bool }
3252         { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3253     }
3254     {
3255         \__enumext_fake_item_viii:
3256         \bool_if:cT { \__enumext_show_length_#1_bool }
3257         { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3258     }
3259 }
3260 }
3261 \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.

```

3262 \NewDocumentEnvironment{enumext}{0}{}
3263 {
3264     \__enumext_safe_exec:
3265     \__enumext_parse_keys:n {#1}
3266     \__enumext_before_list:
3267     \__enumext_start_store_level:
3268     \__enumext_start_list:nn
3269     { \tl_use:c { \__enumext_label_ \__enumext_level: _tl } }
3270     {
3271         \use:c { __enumext_list_arg_two_ \__enumext_level: : }
3272         \__enumext_before_keys_exec:
3273     }
3274     \__enumext_set_item_width:
3275     \__enumext_after_args_exec:
3276 }
3277 {
3278     \__enumext_stop_list:
3279     \__enumext_stop_store_level:
3280     \__enumext_after_list:
3281 }

```

(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.

```

3282 \cs_new_protected:Nn \__enumext_set_item_width:
3283 {
3284     \dim_set:Nn \itemwidth
3285     {
3286         \linewidth
3287     }
3288     \dim_compare:nT
3289     {
3290         \dim_use:c { \__enumext_listoffset_ \__enumext_level: _dim } != \c_zero_dim
3291     }
3292     {
3293         \dim_sub:Nn \itemwidth
3294         {
3295             \dim_use:c { \__enumext_listoffset_ \__enumext_level: _dim }
3296         }
3297     }
3298 }

```

(End of definition for `__enumext_set_item_width:`.)

`__enumext_safe_exec:` The `__enumext_safe_exec:` function first call the function `__enumext_internal_mini_page:` to create the environment `__enumext_mini_env*`, then the function `__enumext_is_not_nested:` which sets `\g__enumext_standar_bool` to “true” if we are not nested within `enumext*`, we will increment `__enumext_level_int` to restrict nesting of the environment, set `__enumext_standar_bool` to “true” and

finally call the function `__enumext_is_on_first_level`: which sets `\l__enumext_standar_first_bool` to “true” only if the environment is not nested and we are at the “first level”.

```

3299 \cs_new_protected:Nn \__enumext_safe_exec:
3300 {
3301   \__enumext_internal_mini_page:
3302   \__enumext_is_not_nested:
3303   \int_incr:N \l__enumext_level_int
3304   \int_compare:nNt { \l__enumext_level_int } > { 4 }
3305   { \msg_fatal:nn { enumext } { list-too-deep } }
3306   \bool_set_true:N \l__enumext_standar_bool
3307   \bool_set_false:N \l__enumext_starred_bool
3308   \__enumext_is_on_first_level:
3309 }

```

(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.

```

3310 \cs_new_protected:Npn \__enumext_parse_keys:n #1
3311 {
3312   \tl_if_novalue:nF {#1}
3313   {
3314     \str_clear:N \l__enumext_series_str
3315     \int_compare:nNtTF { \l__enumext_level_int } = { 1 }
3316     {
3317       \keys_set:nn { enumext / level-1 } {#1}
3318       \__enumext_parse_series:n {#1}
3319       \__enumext_nested_base_line_fix:
3320     }
3321     {
3322       \exp_args:Ne \keys_set:nn
3323       { enumext / level-\int_use:N \l__enumext_level_int } {#1}
3324     }
3325     \__enumext_store_active_keys:n {#1}
3326   }
3327 }

```

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

`__enumext_start_store_level:`

The `__enumext_start_store_level:` and `__enumext_stop_store_level:` functions activate the level saving mechanism for storage in `(sequence)` for the command `\anskey` and the environment `anskey*`.

```

3328 \cs_new_protected:Nn \__enumext_start_store_level:
3329 {
3330   \bool_lazy_all:nT
3331   {
3332     { \bool_if_p:N \l__enumext_store_active_bool }
3333     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3334     { \bool_if_p:N \g__enumext_standar_bool }
3335   }
3336   {
3337     \int_compare:nNt { \l__enumext_level_int } > { 1 }
3338     {
3339       \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3340       \__enumext_store_level_open:
3341     }
3342   }

```

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

```

3343   \bool_lazy_all:nT
3344   {
3345     { \bool_if_p:N \l__enumext_store_active_bool }
3346     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3347     { \int_compare_p:nNt { \l__enumext_level_h_int } = { 1 } }
3348   }
3349   {
3350     \int_compare:nNt { \l__enumext_level_int } > { 0 }
3351     {

```

```

3352         \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3353         \__enumext_store_level_open:
3354     }
3355 }
3356 }

```

Close the stored structure.

```

3357 \cs_new_protected:Nn \__enumext_stop_store_level:
3358 {
3359     \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
3360     {
3361         \__enumext_store_level_close:
3362     }
3363 }

```

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

`__enumext_before_list:` The function `__enumext_before_list:` first calls the function `__enumext_vspace_above:` used by the keys `above` and `above*`, then calls the function `__enumext_before_args_exec:` used by the key `before*` and finally execute the function `__enumext_check_ans_active:` for the check answer mechanism.

```

3364 \cs_new_protected:Nn \__enumext_before_list:
3365 {
3366     \__enumext_vspace_above:
3367     \__enumext_before_args_exec:
3368     \__enumext_check_ans_active:

```

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

```

3369     \dim_compare:nNtT
3370     { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3371     {
3372         \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
3373         {
3374             \linewidth
3375             - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
3376             - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
3377         }

```

The boolean variable `\l__enumext_minipage_active_X_bool` will be activated and the integer variable `\g__enumext_minipage_stat_int` used by the `\miniright` command will be incremented, then the function `__enumext_minipage_add_space:` is called and the `__enumext_mini_env*` environment on the “left side” will be initialized followed by the “vertical spacing” applied to preserve the “baseline” between the *left* and *right* side environments. After these actions, the function `__enumext_multicols_start:` is called to handle the `multicols` environment.

```

3378         \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
3379         \int_gincr:N \g__enumext_minipage_stat_int
3380         \__enumext_minipage_add_space:
3381         \begin{__enumext_mini_env*}
3382         { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
3383     }
3384     \__enumext_multicols_start:
3385 }

```

(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.

```

3386 \cs_new_protected:Nn \__enumext_multicols_start:
3387 {
3388     \int_compare:nNtT
3389     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3390     {
3391         \dim_compare:nNtT
3392         { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3393         {
3394             \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
3395             {

```

```

3396         ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
3397         + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
3398         ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
3399         - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3400     }
3401 }
3402 \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
3403 \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3404 {
3405     \dim_zero:N \columnseprule
3406 }

```

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.

```

3407     \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3408     {
3409         \skip_zero:N \multicolsep
3410         \__enumext_multi_addvspace:
3411     }
3412     \raggedcolumns
3413     \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
3414 }
3415 }

```

(End of definition for `__enumext_multicols_start:`)

`__enumext_multicols_stop:` The function `__enumext_multicols_stop:` will stop the `multicols` environment. If the boolean variable `\l__enumext_minipage_active_X_bool` is false (not nested in `__enumext_mini_env*`) we will apply our “*vertical adjust*” spacing.

```

3416 \cs_new_protected:Nn \__enumext_multicols_stop:
3417 {
3418     \int_compare:nNnT
3419     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3420     {
3421         \end{multicols}
3422         \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3423         {
3424             \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
3425         }
3426     }
3427 }

```

(End of definition for `__enumext_multicols_stop:`)

`__enumext_after_list:` The function `__enumext_after_list:` first check the state of the boolean variable `\l__enumext_minipage_active_X_bool`, if it is “true” a small test will be executed to check if we have omitted the use of `\miniright` (the `__enumext_mini_env*` environment has not been closed), then close `__enumext_mini_env*` and add the *adjusted vertical space* `\l__enumext_minipage_after_skip`, otherwise we will close the `multicols` environment.

```

3428 \cs_new_protected:Nn \__enumext_after_list:
3429 {
3430     \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
3431     {
3432         \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3433         {
3434             \msg_warning:nn { enumext } { missing-miniright }
3435             \miniright
3436         }
3437         \int_gzero:N \g__enumext_minipage_stat_int
3438         \end{\__enumext_mini_env*}
3439         %%% NUEVOOO
3440         %%%\par\addvspace{ 0.5\dp\strutbox }
3441         \int_compare:nNnT
3442         { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } = { 1 }
3443         {
3444             %%\par\addvspace { \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip } }
3445             \par\addvspace{ \l__enumext_minipage_after_skip }
3446             %%\par\par\addvspace{ 0.4\box_dp:N \strutbox }
3447         }
3448     }
3449 }

```

```

3450     \__enumext_multicols_stop:
3451 }

```

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.

```

3452 \__enumext_after_stop_list:
3453 \__enumext_check_ans_key_hook:
3454 \__enumext_vspace_below:
3455 \bool_set_false:N \l__enumext_standar_bool
3456 \__enumext_resume_save_counter:
3457 }

```

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`.

```

3458 \__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.

```

3459 \NewDocumentEnvironment{keyans}{ 0{} }
3460 {
3461   \__enumext_keyans_safe_exec:
3462   \__enumext_keyans_parse_keys:n {#1}
3463   \__enumext_before_list_v:
3464   \__enumext_start_list:nn
3465   { \tl_use:N \l__enumext_label_v_tl }
3466   {
3467     \__enumext_list_arg_two_v:
3468     \__enumext_before_keys_exec_v:
3469   }
3470   \__enumext_keyans_set_item_width:
3471   \__enumext_after_args_exec_v:
3472 }
3473 {
3474   \__enumext_check_starred_cmd:n { item }
3475   \__enumext_stop_list:
3476   \__enumext_after_list_v:
3477 }

```

(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.

```

3478 \cs_new_protected:Nn \__enumext_keyans_set_item_width:
3479 {
3480   \dim_set:Nn \itemwidth
3481   {
3482     \linewidth
3483   }
3484   \dim_compare:nT
3485   {
3486     \l__enumext_listoffset_v_dim != \c_zero_dim
3487   }
3488   {
3489     \dim_sub:Nn \itemwidth
3490     {
3491       \l__enumext_listoffset_v_dim
3492     }
3493   }
3494 }

```

(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.

```

3495 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3496 {
3497   \bool_if:NF \l__enumext_store_active_bool
3498   {
3499     \msg_error:nnnn { enumext } { wrong-place } { keyans } { save-ans }
3500   }
3501   \int_incr:N \l__enumext_keyans_level_int
3502   \bool_set_true:N \l__enumext_keyans_env_bool
3503   \__enumext_keyans_name_and_start:
3504   % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3505   \bool_set_false:N \l__enumext_store_active_bool
3506   \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
3507   {
3508     \msg_error:nn { enumext } { keyans-nested }
3509   }
3510   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3511   {
3512     \msg_error:nn { enumext } { keyans-wrong-level }
3513   }
3514 }

```

(End of definition for `__enumext_keyans_safe_exec:`)

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

```

3515 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3516 {
3517   \keys_set:nn { enumext / keyans } {#1}
3518 }

```

(End of definition for `__enumext_keyans_parse_keys:n`)

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

```

\__enumext_keyans_multicols_start:
\__enumext_keyans_multicols_stop:
\__enumext_after_list_v:
3519 \cs_new_protected:Nn \__enumext_before_list_v:
3520 {
3521   \__enumext_vspace_above_v:
3522   \__enumext_before_args_exec_v:
3523   \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3524   {
3525     \dim_set:Nn \l__enumext_minipage_left_v_dim
3526     {
3527       \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3528     }
3529     \bool_set_true:N \l__enumext_minipage_active_v_bool
3530     \int_gincr:N \g__enumext_minipage_stat_int
3531     \__enumext_keyans_mini_addvspace:
3532     \nointerlineskip\noindent
3533     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
3534   }
3535   \__enumext_keyans_multicols_start:
3536 }
3537 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3538 {
3539   \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3540   {
3541     \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3542     {
3543       \dim_set:Nn \l__enumext_columns_sep_v_dim
3544       {
3545         (
3546           \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3547         ) / \l__enumext_columns_v_int
3548         - \l__enumext_listoffset_v_dim
3549       }
3550     }
3551     \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim
3552     \skip_zero:N \multicolsep
3553     \dim_zero:N \columnseprule % no rule here
3554     \bool_if:NF \l__enumext_minipage_active_v_bool

```

```
3555         {
3556             \__enumext_keyans_multi_addvspace:
3557         }
3558         \raggedcolumns
3559         \begin{multicols}{\l__enumext_columns_v_int }
3560     }
3561 }
3562 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3563 {
3564     \int_compare:nNt { \l__enumext_columns_v_int } > { 1 }
3565     {
3566         \end{multicols}
3567         \bool_if:NF \l__enumext_minipage_active_v_bool
3568         {
3569             \par\addvspace{ \l__enumext_multicols_below_v_skip }
3570         }
3571     }
3572 }
3573 \cs_new_protected:Nn \__enumext_after_list_v:
3574 {
3575     \bool_if:NTF \l__enumext_minipage_active_v_bool
3576     {
3577         \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
3578         {
3579             \msg_warning:nn { enumext } { missing-miniright }
3580             \miniright
3581         }
3582         \int_gzero:N \g__enumext_minipage_stat_int
3583         \end{__enumext_mini_env*}
3584         \par\addvspace{ \l__enumext_minipage_after_skip }
3585     }
3586     {
3587         \__enumext_keyans_multicols_stop:
3588     }
3589     \bool_set_false:N \l__enumext_keyans_env_bool
3590     \__enumext_after_stop_list_v:
3591     \__enumext_vspace_below_v:
3592 }
```

(End of definition for __enumext_before_list_v: and others.)

12.40 The environment keyanspic and \anspic

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

The contents are passed to the environment by means of the `\anspic` command and are placed inside `minipage` environments, with the *<label>* underneath, adjusting widths according to the options passed to the environment.

Again it is necessary to “adjust” the spacing, both vertical and horizontal, to obtain an output like the one shown in the figure 12.

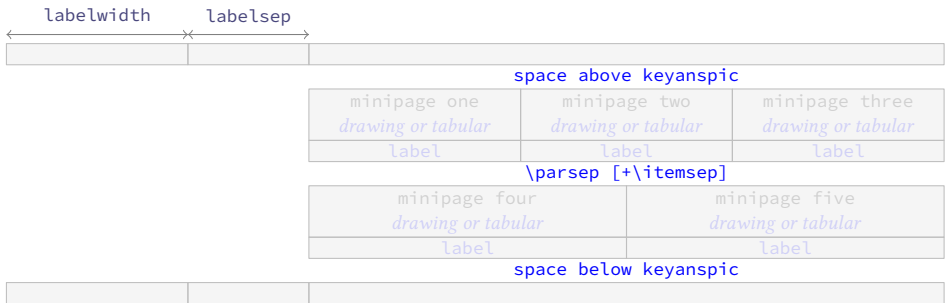


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

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

12.40.1 The command \anspic

`\anspic` The `\anspic` command take three arguments, the starred (*) versions `\anspic*` and `\anspic* [<content>]` store the current *<label>* next to the `[<content>]` 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.

```
3593 \NewDocumentCommand \anspic { s o +m }
3594 {
```


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

```

3595 \bool_if:NF \__enumext_store_active_bool
3596 {
3597   \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
3598 }
3599 \int_compare:nNt { \__enumext_level_int } > { 1 }
3600 {
3601   \msg_error:nn { enumext } { keyanspic-wrong-level }
3602 }
3603 \int_compare:nNt { \__enumext_keyans_level_int } = { 1 }
3604 {
3605   \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
3606 }

```

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.

```

3607 \seq_put_right:Nn \__enumext_keyans_pic_body_seq
3608 {
3609   \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
3610 }
3611 }

```

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

`__enumext_keyans_anspic_code:nnn`

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

```

3612 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
3613 {
3614   \stepcounter { enumXvi }
3615   #3 \\\
3616   \bool_if:nT { #1 }
3617   {
3618     \__enumext_keyans_addto_prop:n { #2 }
3619     \__enumext_keyans_store_ref:
3620     \__enumext_keyans_addto_seq:n { #2 }
3621     \int_gincr:N \__enumext_check_starred_cmd_int
3622     \bool_lazy_or:nnT
3623     { \bool_if_p:N \__enumext_show_answer_bool }
3624     { \bool_if_p:N \__enumext_show_position_bool }
3625     {
3626       \tl_set_eq:NN \__enumext_label_v_tl \__enumext_label_vi_tl
3627       \__enumext_keyans_show_left:n { #2 }
3628       \tl_set_eq:NN \__enumext_label_vi_tl \__enumext_label_v_tl
3629     }
3630   }
3631   \tl_use:N \__enumext_label_font_style_v_tl
3632   \__enumext_wrapper_label_v:n { \__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
3633 }

```

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

12.40.2 The environment `keyanspic`

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

```

3634 \NewDocumentEnvironment{keyanspic}{o}
3635 {
3636   \__enumext_keyans_pic_safe_exec:
3637   \__enumext_start_list:nn
3638   { }
3639   {
3640     \__enumext_keyans_pic_arg_two:
3641   }

```

We apply the “adjusted” vertical spacing above the environment

```

3642 \vspace { \__enumext_keyans_pic_above_skip }
3643 }

```

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

```

3644 {
3645   \tl_if_novalue:nTF { #1 }
3646   {
3647     \__enumext_keyans_pic_do:e { \seq_count:N \__enumext_keyans_pic_body_seq }
3648   }
3649   { \__enumext_keyans_pic_do:n { #1 } }
3650   \__enumext_stop_list:
3651   \__enumext_check_starred_cmd:n { anspic }
3652   \setcounter { enumXvi } { 0 }
3653   \vspace { \__enumext_topsep_v_skip }
3654   %\bool_set_false:N \__enumext_store_active_bool
3655 }

```

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

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

```

3656 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
3657 {
3658   \int_incr:N \__enumext_keyans_pic_level_int
3659   \int_compare:nNnT { \__enumext_keyans_pic_level_int } > { 1 }
3660   {
3661     \msg_error:nn { enumext } { keyanspic-nested }
3662   }
3663   \__enumext_keyans_name_and_start:
3664 }

```

(End of definition for `__enumext_keyans_pic_safe_exec:`.)

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

```

3665 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3666 {
3667   \dim_compare:nNnT { #1 } < { 0pt }
3668   { \skip_set:Nn #1 { -#1 } }
3669 }

```

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

`__enumext_keyans_pic_arg_two:` The function `__enumext_keyans_pic_arg_two:` will be used in the second argument of the `__enumext_start_list:nn` function that defines the `keyanspic` environment, it will handle the setting of spaces.

```

3670 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
3671 {

```

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

```

3672   \bool_set_false:N \__enumext_leftmargin_tmp_v_bool
3673   \__enumext_list_arg_two_v:

```

We will add the value of `\itemsep` to `\parsep` which we will use as vertical spacing between the above and below `minipage` environments. and adjust the value of `\leftmargin`, the label and counter are handled directly by the `\anspic` command. Then we make equal to zero `\labelwidth`, `\labelsep`, `\partopsep` and `\itemsep` so that the horizontal and vertical spacing is not affected.

```

3674   \skip_add:Nn \parsep { \itemsep }
3675   \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
3676   \dim_zero:N \labelwidth
3677   \dim_zero:N \listparindent
3678   \dim_zero:N \labelsep
3679   \skip_zero:N \partopsep
3680   \skip_zero:N \itemsep

```

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

```

3681   \__enumext_keyans_pic_skip_abs:N \parsep
3682   \skip_set:Nn \__enumext_keyans_pic_above_skip
3683   {

```

```

3684     \box_dp:N \strutbox
3685     + \l__enumext_topsep_v_skip
3686     - \parsep
3687   }
3688   \__enumext_item_std:w \scan_stop:
3689   % paranoia
3690   \RenewDocumentCommand \item {}
3691   {
3692     \msg_error:nn { enumext } { keyanspic-item-cmd }
3693   }
3694 }

```

(End of definition for `__enumext_keyans_pic_arg_two:.`)

```

\__enumext_keyans_pic_do:n
\__enumext_keyans_pic_do:e

```

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

```

3695 \cs_new_protected:Nn \__enumext_keyans_pic_do:n
3696 {
3697   \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
3698 }
3699 \cs_generate_variant:Nn \__enumext_keyans_pic_do:n { e }

```

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

```

\__enumext_keyans_pic_row:n

```

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

```

3700 \cs_new_protected:Nn \__enumext_keyans_pic_row:n
3701 {
3702   \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
3703   \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
3704   \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
3705   \int_step_inline:nnn
3706   { \l__enumext_keyans_pic_above_int + 1 }
3707   { \l__enumext_keyans_pic_below_int }
3708   {
3709     \__enumext_minipage:w [ b ]{ \l__enumext_keyans_pic_width_dim }
3710     \centering
3711     \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
3712     \__enumext_endminipage:
3713   }
3714   \par
3715 }

```

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

12.41 The horizontal environments

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

12.42 Redefining `\footnote` command

```

\__enumext_footnotetext:nn
\__enumext_renew_footnote:
\__enumext_print_footnote:

```

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

```

3716 \cs_new_protected:Nn \__enumext_footnotetext:nn
3717 {
3718   \footnotetext[#1]{#2}
3719 }
3720 \cs_new_protected:Nn \__enumext_renew_footnote:
3721 {
3722   \seq_gclear:N \g__enumext_footnote_arg_seq
3723   \seq_gclear:N \g__enumext_footnote_int_seq
3724   \RenewDocumentCommand \footnote { o +m }
3725   {
3726     \tl_if_novalue:nTF {##1}
3727     {
3728       \stepcounter{footnote}
3729       \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
3730     }

```

```

3731     {
3732         \int_gset:Nn \g__enumext_footnote_int { ##1 }
3733     }
3734     \footnotemark [ \g__enumext_footnote_int ]
3735     \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
3736     \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
3737 }
3738 }
3739 \cs_new_protected:Nn \__enumext_print_footnote:
3740 {
3741     \seq_if_empty:NF \g__enumext_footnote_int_seq
3742     {
3743         \seq_map_pairwise_function:NNN
3744         \g__enumext_footnote_int_seq
3745         \g__enumext_footnote_arg_seq
3746         \__enumext_footnotetext:nn
3747     }
3748 }

```

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

12.42.1 Functions for item box width

To achieve the horizontal list environment we will capture the `\item` command and the content of this in an plain `lrbox` box using `\makebox` for the `label` and a `minipage` environment for the content passed to `\item`, we will also add the optional argument (`\langle number \rangle`) to `\item` to be able to *join columns* horizontally, in simple terms, we want `\item` to behave in the same way as in the `enumext` environment but adding an optional first argument (`\langle number \rangle`).

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

`__enumext_starred_columns_set_vii:`
`__enumext_starred_columns_set_viii:`

```

3749 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
3750 {
3751     \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3752     {
3753         \dim_set:Nn \l__enumext_columns_sep_vii_dim
3754         {
3755             ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
3756             / \l__enumext_columns_vii_int
3757         }
3758     }
3759     \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - 1 }
3760     \dim_set:Nn \l__enumext_item_width_vii_dim
3761     {
3762         ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
3763         / \l__enumext_columns_vii_int
3764         - \l__enumext_labelwidth_vii_dim
3765         - \l__enumext_labelsep_vii_dim
3766     }

```

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

```

3767     \dim_compare:nNnT { \l__enumext_rightmargin_vii_dim } > { \c_zero_dim }
3768     {
3769         \dim_sub:Nn \l__enumext_item_width_vii_dim
3770         {
3771             ( \l__enumext_rightmargin_vii_dim * \l__enumext_tmpa_vii_int )
3772             / \l__enumext_columns_vii_int
3773         }
3774         \dim_add:Nn \l__enumext_columns_sep_vii_dim
3775         {
3776             \l__enumext_rightmargin_vii_dim
3777         }
3778     }
3779 }

```

Same implementation for the `keyans*` environment.

```

3780 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3781 {
3782     \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3783     {
3784         \dim_set:Nn \l__enumext_columns_sep_viii_dim
3785         {
3786             ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
3787             / \l__enumext_columns_viii_int

```

```

3788     }
3789   }
3790   \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - 1 }
3791   \dim_set:Nn \l__enumext_item_width_viii_dim
3792   {
3793     ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
3794     / \l__enumext_columns_viii_int
3795     - \l__enumext_labelwidth_viii_dim
3796     - \l__enumext_labelsep_viii_dim
3797   }
3798   \dim_compare:nNnT { \l__enumext_rightmargin_viii_dim } > { \c_zero_dim }
3799   {
3800     \dim_sub:Nn \l__enumext_item_width_viii_dim
3801     {
3802       ( \l__enumext_rightmargin_viii_dim * \l__enumext_tmpa_vii_int )
3803       / \l__enumext_columns_viii_int
3804     }
3805     \dim_add:Nn \l__enumext_columns_sep_viii_dim
3806     {
3807       \l__enumext_rightmargin_viii_dim
3808     }
3809   }
3810 }

```

(End of definition for `__enumext_starred_columns_set_vii:` and `__enumext_starred_columns_set_viii:`)

12.42.2 Functions for join item columns

`__enumext_starred_joined_item_vii:n`
`__enumext_starred_joined_item_viii:n`

The functions `__enumext_starred_joined_item_vii:n` and `__enumext_starred_joined_item_viii:n` will set the *width* of the box in which the content passed to `\item(columns)` will be stored together with the value of `\itemwidth` for the `enumext*` environment.

```

3811 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
3812 {
3813   \int_set:Nn \l__enumext_joined_item_vii_int {#1}
3814   \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
3815   {
3816     \msg_warning:nnee { enumext } { item-joined }
3817     { \int_use:N \l__enumext_joined_item_vii_int }
3818     { \int_use:N \l__enumext_columns_vii_int }
3819     \int_set:Nn \l__enumext_joined_item_vii_int
3820     {
3821       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
3822     }
3823   }
3824   \int_compare:nNnT
3825   { \l__enumext_joined_item_vii_int }
3826   >
3827   { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
3828   {
3829     \msg_warning:nnee { enumext } { item-joined-columns }
3830     { \int_use:N \l__enumext_joined_item_vii_int }
3831     {
3832       \int_eval:n
3833       { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
3834     }
3835     \int_set:Nn \l__enumext_joined_item_vii_int
3836     {
3837       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
3838     }
3839   }
3840   \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { 1 }
3841   {
3842     \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
3843     \int_decr:N \l__enumext_joined_item_aux_vii_int
3844     \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3845     \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3846     \dim_set:Nn \l__enumext_joined_width_vii_dim
3847     {
3848       \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
3849       + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3850         + \l__enumext_columns_sep_vii_dim
3851         ) * \l__enumext_joined_item_aux_vii_int

```

```

3852     }
3853     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
3854 }
3855 {
3856     \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
3857     \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
3858 }
3859 }

```

Same implementation for the `keyans*` environment.

```

3860 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
3861 {
3862     \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3863     \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3864     {
3865         \msg_warning:nnee { enumext } { item-joined }
3866         { \int_use:N \l__enumext_joined_item_viii_int }
3867         { \int_use:N \l__enumext_columns_viii_int }
3868         \int_set:Nn \l__enumext_joined_item_viii_int
3869         {
3870             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
3871         }
3872     }
3873     \int_compare:nNnT
3874     { \l__enumext_joined_item_viii_int }
3875     >
3876     { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
3877     {
3878         \msg_warning:nnee { enumext } { item-joined-columns }
3879         { \int_use:N \l__enumext_joined_item_viii_int }
3880         {
3881             \int_eval:n
3882             { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
3883         }
3884         \int_set:Nn \l__enumext_joined_item_viii_int
3885         {
3886             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
3887         }
3888     }
3889     \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { 1 }
3890     {
3891         \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3892         \int_decr:N \l__enumext_joined_item_aux_viii_int
3893         \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
3894         \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
3895         \dim_set:Nn \l__enumext_joined_width_viii_dim
3896         {
3897             \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3898             + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3899               + \l__enumext_columns_sep_viii_dim
3900               ) * \l__enumext_joined_item_aux_viii_int
3901         }
3902         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3903     }
3904     {
3905         \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3906         \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3907     }
3908 }

```

(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

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

```

3909 \cs_new_protected:Nn \__enumext_start_mini_vii:
3910 {
3911     \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
3912     {

```

```

3913     \dim_set:Nn \l__enumext_minipage_left_vii_dim
3914     {
3915         \linewidth
3916         - \l__enumext_minipage_right_vii_dim
3917         - \l__enumext_minipage_hsep_vii_dim
3918     }
3919     \bool_set_true:N \l__enumext_minipage_active_vii_bool
3920     \dim_gset_eq:NN
3921         \g__enumext_minipage_right_vii_dim
3922         \l__enumext_minipage_right_vii_dim
3923     \__enumext_mini_addvspace_vii:
3924     \nointerlineskip\noindent
3925     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
3926 }
3927 }

```

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

```

3928 \cs_new_protected:Nn \__enumext_stop_mini_vii:
3929 {
3930     \bool_if:NT \l__enumext_minipage_active_vii_bool
3931     {
3932         \end{__enumext_mini_env*}
3933         \hfill
3934         \bool_gset_true:N \g__enumext_minipage_active_vii_bool
3935     }
3936 }

```

Finally we execute the `{\code}` passed to the `mini-right` or `mini-right*` keys stored in the variable `\g__enumext_miniright_code_vii_tl` in the `__enumext_mini_env*` environment on the “right side”. For compatibility with the `caption` package and possibly other `{\code}` passed to this key, we will pass it to a box and then print it.

```

3937 \__enumext_after_env:nn {enumext*}
3938 {
3939     \bool_if:NT \g__enumext_minipage_active_vii_bool
3940     {
3941         \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
3942         \par\addvspace { \g__enumext_minipage_right_skip }
3943         \bool_if:NF \g__enumext_minipage_center_vii_bool
3944         {
3945             \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
3946             {
3947                 \centering
3948             }
3949         }
3950         \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
3951         {
3952             \tl_use:N \g__enumext_miniright_code_vii_tl
3953         }
3954         \box_use_drop:N \l__enumext_miniright_code_vii_box
3955         \end{__enumext_mini_env*}
3956         \par\addvspace{ \g__enumext_minipage_after_skip }
3957     }
3958     \bool_gset_false:N \g__enumext_minipage_active_vii_bool
3959     \bool_gset_true:N \g__enumext_minipage_center_vii_bool
3960     \tl_gclear:N \g__enumext_miniright_code_vii_tl
3961     \dim_gzero:N \g__enumext_minipage_right_vii_dim
3962     \bool_gset_false:N \g__enumext_starred_bool
3963 }

```

(End of definition for `__enumext_start_mini_vii:` and `__enumext_stop_mini_vii:`)

`__enumext_start_mini_viii:` The implementation of the `mini-env`, `mini-right` and `mini-right*` keys is identical to the one used in the `enumext*` environment.

```

3964 \cs_new_protected:Nn \__enumext_start_mini_viii:
3965 {
3966     \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3967     {
3968         \dim_set:Nn \l__enumext_minipage_left_viii_dim

```



```

3969     {
3970         \linewidth
3971         - \l__enumext_minipage_right_viii_dim
3972         - \l__enumext_minipage_hsep_viii_dim
3973     }
3974     \bool_set_true:N \l__enumext_minipage_active_viii_bool
3975     \dim_gset_eq:NN
3976         \g__enumext_minipage_right_viii_dim
3977         \l__enumext_minipage_right_viii_dim
3978     \__enumext_mini_addvspace_viii:
3979     \nointerlineskip\noindent
3980     \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
3981     }
3982 }
3983 \cs_new_protected:Nn \__enumext_stop_mini_viii:
3984 {
3985     \bool_if:NT \l__enumext_minipage_active_viii_bool
3986     {
3987         \end{__enumext_mini_env*}
3988         \hfill
3989         \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3990     }
3991 }
3992 \__enumext_after_env:n {keyans*}
3993 {
3994     \bool_if:NT \g__enumext_minipage_active_viii_bool
3995     {
3996         \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3997         \par\addvspace { \g__enumext_minipage_right_skip }
3998         \bool_if:NF \g__enumext_minipage_center_viii_bool
3999         {
4000             \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
4001             {
4002                 \centering
4003             }
4004         }
4005         \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
4006         {
4007             \tl_use:N \g__enumext_miniright_code_viii_tl
4008         }
4009         \box_use_drop:N \l__enumext_miniright_code_viii_box
4010         \end{__enumext_mini_env*}
4011         \par\addvspace{ \g__enumext_minipage_after_skip }
4012     }
4013     \bool_gset_false:N \g__enumext_minipage_active_viii_bool
4014     \bool_gset_true:N \g__enumext_minipage_center_viii_bool
4015     \tl_gclear:N \g__enumext_miniright_code_viii_tl
4016     \dim_gzero:N \g__enumext_minipage_right_viii_dim
4017 }

```

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

12.43 The environment enumext*

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

```

4018 \NewDocumentEnvironment{enumext*}{ o }
4019 {
4020     \__enumext_safe_exec_vii:
4021     \__enumext_parse_keys_vii:n {#1}
4022     \__enumext_before_list_vii:
4023     \__enumext_start_store_level_vii:
4024     \__enumext_start_list:n { }
4025     {
4026         \__enumext_list_arg_two_vii:
4027         \__enumext_before_keys_exec_vii:
4028     }
4029     \__enumext_starred_columns_set_vii:
4030     \item[] \scan_stop:
4031     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
4032     \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:

```

```

4033     }
4034     {
4035         \__enumext_stop_item_tmp_vii:
4036         \__enumext_remove_extra_parsep_vii:
4037         \__enumext_stop_list:
4038         \__enumext_stop_store_level_vii:
4039         \__enumext_after_list_vii:
4040     }

```

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

`__enumext_safe_exec_vii:` We will first call the function `__enumext_internal_mini_page:` to create the environment `__enumext-mini-env*`, then the function `__enumext_is_not_nested:` which sets `\g__enumext_starred_bool` to true if we are not nested within `enumext`, we will increment `\l__enumext_level_h_int` to restrict nesting of the environment, set `\l__enumext_starred_bool` to true and finally call the function `__enumext_is_on_first_level:` which sets `\l__enumext_starred_first_bool` to true if we are not nested, allowing the “storage system” to be used.

```

4041 \cs_new_protected:Nn \__enumext_safe_exec_vii:
4042 {
4043     \__enumext_internal_mini_page:
4044     \__enumext_is_not_nested:
4045     \int_incr:N \l__enumext_level_h_int
4046     \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
4047     {
4048         \msg_error:nn { enumext } { nested }
4049     }
4050     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
4051     {
4052         \msg_error:nnn { enumext } { nested-horizontal } { keyans* }
4053     }
4054     \bool_set_true:N \l__enumext_starred_bool
4055     \bool_set_false:N \l__enumext_standar_bool
4056     \__enumext_is_on_first_level:
4057 }

```

(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`.

```

4058 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
4059 {
4060     \tl_if_novalue:nF {#1}
4061     {
4062         \str_clear:N \l__enumext_series_str
4063         \keys_set:nn { enumext / enumext* } {#1}
4064         \__enumext_parse_series:n {#1}
4065         \__enumext_store_active_keys_vii:n {#1}
4066         \__enumext_nested_base_line_fix:
4067     }
4068 }

```

(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*`.

```

4069 \cs_new_protected:Nn \__enumext_before_list_vii:
4070 {
4071     \__enumext_vspace_above_vii:
4072     \__enumext_check_ans_active:
4073     \__enumext_before_args_exec_vii:
4074     \__enumext_start_mini_vii:
4075 }

```

(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 `__enumext_starred_bool` to false and call the `__enumext_resume_save_counter:` function used by the `series`, `resume` and `resume*` keys.

```

4076 \cs_new_protected:Nn \__enumext_after_list_vii:
4077 {
4078   \__enumext_stop_mini_vii:
4079   \__enumext_after_stop_list_vii:
4080   \__enumext_check_ans_key_hook:
4081   \__enumext_vspace_below_vii:
4082   \bool_set_false:N \__enumext_starred_bool
4083   \__enumext_resume_save_counter:
4084 }

```

(End of definition for `__enumext_after_list_vii:`.)

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

`__enumext_stop_store_level_vii:`

```

4085 \cs_new_protected:Nn \__enumext_start_store_level_vii:
4086 {
4087   \bool_if:NT \__enumext_store_active_bool
4088   {
4089     \int_compare:nNnT { \__enumext_level_int } > { 0 }
4090     {
4091       \__enumext_store_level_open_vii:
4092     }
4093   }
4094 }
4095 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
4096 {
4097   \bool_if:NT \__enumext_store_active_bool
4098   {
4099     \int_compare:nNnT { \__enumext_level_int } > { 0 }
4100     {
4101       \__enumext_store_level_close_vii:
4102     }
4103   }
4104 }

```

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

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

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

```

4105 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
4106 {
4107   \__enumext_stop_item_tmp_vii:
4108   \int_incr:N \__enumext_item_column_pos_vii_int
4109   \int_gincr:N \g__enumext_item_count_all_vii_int
4110   \__enumext_item_peek_args_vii:
4111 }

```

(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).

```

4112 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
4113 {
4114   \peek_meaning:NTF (
4115     { \__enumext_joined_item_vii:w }
4116     { \__enumext_joined_item_vii:w (1) }
4117   }

```

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

`__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`.

```
4118 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
4119 {
4120   \__enumext_starred_joined_item_vii:n {#1}
4121   \peek_meaning_remove:NTF *
4122   { \__enumext_starred_item_vii:w }
4123   { \__enumext_standar_item_vii:w }
4124 }
```

(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]`.

```
4125 \cs_new_protected:Npn \__enumext_standar_item_vii:w
4126 {
4127   \bool_set_false:N \l__enumext_item_starred_vii_bool
4128   \peek_meaning:NTF [
4129   {
4130     \bool_set_eq:NN
4131     \l__enumext_wrap_label_vii_bool
4132     \l__enumext_wrap_label_opt_vii_bool
4133     \__enumext_start_item_vii:w
4134   }
4135   {
4136     \bool_set_true:N \l__enumext_wrap_label_vii_bool
4137     \legacy_if_set_true:n { @noitemarg }
4138     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4139   }
4140 }
```

(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`

```
4141 \cs_new_protected:Npn \__enumext_starred_item_vii:w
4142 {
4143   \bool_set_true:N \l__enumext_item_starred_vii_bool
4144   \bool_set_true:N \l__enumext_wrap_label_vii_bool
4145   \peek_meaning:NTF [
4146   { \__enumext_starred_item_vii_aux_i:w }
4147   { \__enumext_starred_item_vii_aux_ii:w }
4148 }
4149 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
4150 {
4151   \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
4152   \__enumext_starred_item_vii_aux_ii:w
4153 }
4154 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
4155 {
4156   \peek_meaning:NTF [
4157   { \__enumext_starred_item_vii_aux_iii:w }
4158   {
4159     \dim_set_eq:NN
4160     \l__enumext_item_symbol_sep_vii_dim
4161     \l__enumext_labelsep_vii_dim
4162     \legacy_if_set_true:n { @noitemarg }
4163     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4164   }
4165 }
4166 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
```

```

4167 {
4168   \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
4169   \legacy_if_set_true:n { @noitemarg }
4170   \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4171 }

```

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

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

`__enumext_start_item_vii:w`

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

```

4172 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
4173 {
4174   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
4175   \legacy_if:nT { @noitemarg }
4176   {
4177     \legacy_if_set_false:n { @noitemarg }
4178     \legacy_if:nT { @nmbrrlist }
4179     {
4180       \bool_if:NT \l__enumext_hyperref_bool
4181       {
4182         \legacy_if_set_true:n { @hyper@item }
4183       }
4184       \refstepcounter{enumXvii}
4185       \bool_if:NT \l__enumext_check_answers_bool
4186       {
4187         \int_gincr:N \g__enumext_item_number_int
4188         \bool_set_true:N \l__enumext_item_number_bool
4189       }
4190     }
4191   }

```

Here we start capturing `\item` and its contents into a group using the plain form of the `lrbox` environment. If the state of the variable `\l__enumext_footnotes_key_bool` is false, we will redefine the command `\footnote`, followed by printing the $\langle symbol \rangle$ defined for `\item*` if it is present and open a new group inside which we execute `font key` next to `\item` and the keys `wrap-label`, `wrap-label*`, `align`, close the group and execute the key `labelsep` and then the key `first`. Finally we open the `minipage` environment and execute the `listparindent` key which will be equal to `\parindent`, the `parsep` key which will be equal to `\parskip` and the `itemindent` key.

```

4192 \group_begin:
4193   \lrbox{ \l__enumext_item_text_vii_box }
4194   \bool_if:NF \l__enumext_footnotes_key_bool
4195   {
4196     \__enumext_renew_footnote:
4197   }
4198   \bool_if:NT \l__enumext_item_starred_vii_bool
4199   {
4200     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
4201     {
4202       \tl_gset_eq:NN
4203       \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
4204     }
4205     \mode_leave_vertical:
4206     \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
4207     \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
4208     \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
4209     \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
4210   }
4211   \group_begin:
4212     \tl_use:N \l__enumext_label_font_style_vii_tl
4213     \bool_if:NTF \l__enumext_wrap_label_vii_bool
4214     {
4215       \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
4216       { \__enumext_wrapper_label_vii:n {#1} }
4217     }

```

```

4218         {
4219             \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
4220         }
4221     \group_end:
4222     \skip_horizontal:N \l__enumext_labelsep_vii_dim
4223     \tl_use:N \l__enumext_after_list_args_vii_tl
4224     \l__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
4225     \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
4226     \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
4227     \tl_use:N \l__enumext_fake_item_indent_vii_tl
4228 }

```

(End of definition for `\l__enumext_start_item_vii:w`.)

`\l__enumext_stop_item_vii:` The function `\l__enumext_stop_item_vii:` shall terminate with the capture of `\item` and its *contents*. Close the environments `minipage`, `lrbox` and the group. Then we only have to set the width of the box and print it next to `\footnote`, and add the horizontal and vertical separation between the boxes.

```

4229 \cs_new_protected_nopar:Nn \l__enumext_stop_item_vii:
4230 {
4231     \l__enumext_endminipage:
4232     \endlrbox
4233     \group_end:
4234     \box_set_wd:Nn \l__enumext_item_text_vii_box
4235     {
4236         \l__enumext_joined_width_vii_dim
4237         + \l__enumext_labelwidth_vii_dim
4238         + \l__enumext_labelsep_vii_dim
4239     }
4240     \int_set:Nn \hbadness { 10000 }
4241     \box_use_drop:N \l__enumext_item_text_vii_box
4242     \bool_if:NF \l__enumext_footnotes_key_bool
4243     {
4244         \l__enumext_print_footnote:
4245     }
4246     \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
4247     {
4248         \par\noindent
4249         \int_zero:N \l__enumext_item_column_pos_vii_int
4250     }
4251     { \hspace{ \l__enumext_columns_sep_vii_dim } }
4252 }

```

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

`\l__enumext_remove_extra_parsep_vii:` Finally we will remove the vertical space equal to `\parsep` when the total number of items is divisible by the number of items in the last row of the environment.

```

4253 \cs_new_protected:Nn \l__enumext_remove_extra_parsep_vii:
4254 {
4255     \int_compare:nNnTF
4256     {
4257         \int_mod:nn { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
4258     }
4259     =
4260     { 0 }
4261     {
4262         \par
4263         \vspace{ -\l__enumext_itemsep_vii_skip }
4264         \int_gzero:N \g__enumext_item_count_all_vii_int
4265     }
4266 }

```

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 `\l__enumext_after_env:nn`.

```

4267 \l__enumext_after_env:nn {enumext*} { \l__enumext_execute_after_env: }

```

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

12.44 The environment keyans*

keyans*

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

```

4268 \NewDocumentEnvironment{keyans*}{o}{
4269   {
4270     \__enumext_safe_exec_viii:
4271     \__enumext_parse_keys_viii:n {#1}
4272     \__enumext_before_list_viii:
4273     \__enumext_start_list:nn { }
4274     {
4275       \__enumext_list_arg_two_viii:
4276       \__enumext_before_keys_exec_viii:
4277     }
4278     \__enumext_starred_columns_set_viii:
4279     \item[] \scan_stop:
4280     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
4281     \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
4282   }
4283   {
4284     \__enumext_stop_item_tmp_viii:
4285     \__enumext_remove_extra_parsep_viii:
4286     \__enumext_check_starred_cmd:n { \item }
4287     \__enumext_stop_list:
4288     \__enumext_after_list_viii:
4289   }

```

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

__enumext_safe_exec_viii:

First check the maximum nesting level for the keyans* environment.

```

4290 \cs_new_protected:Nn \__enumext_safe_exec_viii:
4291   {
4292     \int_incr:N \__enumext_keyans_level_h_int
4293     \int_compare:nNnT { \__enumext_keyans_level_h_int } > { 1 }
4294     {
4295       \msg_error:nn { enumext } { nested }
4296     }
4297     \__enumext_keyans_name_and_start:
4298     \bool_if:NT \__enumext_starred_bool
4299     {
4300       \msg_error:nnn { enumext } { nested-horizontal } { enumext* }
4301     }
4302     \bool_set_true:N \__enumext_starred_bool
4303     % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
4304     \bool_set_false:N \__enumext_store_active_bool
4305     \int_compare:nNnT { \__enumext_level_int } > { 1 }
4306     {
4307       \msg_error:nn { enumext } { keyans-wrong-level }
4308     }
4309   }

```

(End of definition for __enumext_safe_exec_viii:.)

__enumext_parse_keys_viii:n

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

```

4310 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
4311   {
4312     \tl_if_novalue:nF {#1}
4313     {
4314       \keys_set:nn { enumext / keyans* } {#1}
4315     }
4316   }

```

(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`.

```

4317 \cs_new_protected:Nn \__enumext_before_list_viii:
4318   {
4319     \__enumext_vspace_above_viii:

```



```

4320     \__enumext_before_args_exec_viii:
4321     \__enumext_start_mini_viii:
4322 }

```

(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.

```

4323 \cs_new_protected:Nn \__enumext_after_list_viii:
4324 {
4325     \__enumext_stop_mini_viii:
4326     \__enumext_after_stop_list_viii:
4327     \__enumext_vspace_below_viii:
4328 }

```

(End of definition for __enumext_after_list_viii:.)

12.44.1 The command \item in keyans*

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

__enumext_start_item_tmp_viii: First we will call the function __enumext_stop_item_tmp_viii: that we will redefine later, we will increment the value of __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.

```

4329 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
4330 {
4331     \__enumext_stop_item_tmp_viii:
4332     \int_incr:N \__enumext_item_column_pos_viii_int
4333     \int_gincr:N \g__enumext_item_count_all_viii_int
4334     \__enumext_item_peek_args_viii:
4335 }

```

(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).

```

4336 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
4337 {
4338     \peek_meaning:NTF (
4339     { \__enumext_joined_item_viii:w }
4340     { \__enumext_joined_item_viii:w (1) }
4341 }

```

(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.

```

4342 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
4343 {
4344     \__enumext_starred_joined_item_viii:n {#1}
4345     \peek_meaning_remove:NTF *
4346     { \__enumext_starred_item_viii:w }
4347     { \__enumext_standar_item_viii:w }
4348 }

```

(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]`.

```

4349 \cs_new_protected:Npn \__enumext_standar_item_viii:w
4350 {
4351   \bool_set_false:N \l__enumext_item_starred_viii_bool
4352   \peek_meaning:NTF [
4353     {
4354       \bool_set_eq:NN
4355         \l__enumext_wrap_label_viii_bool
4356         \l__enumext_wrap_label_opt_viii_bool
4357       \__enumext_start_item_viii:w
4358     }
4359     {
4360       \bool_set_true:N \l__enumext_wrap_label_viii_bool
4361       \legacy_if_set_true:n { @noitemarg }
4362       \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4363     }
4364   }

```

(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*[⟨content⟩]`.

```

4365 \cs_new_protected:Npn \__enumext_starred_item_viii:w
4366 {
4367   \bool_set_true:N \l__enumext_item_starred_viii_bool
4368   \bool_set_true:N \l__enumext_wrap_label_viii_bool
4369   \peek_meaning:NTF [
4370     { \__enumext_starred_item_viii_aux_i:w }
4371     { \__enumext_starred_item_viii_aux_ii:w }
4372   }

```

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`.

```

4373 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
4374 {
4375   \tl_clear:N \l__enumext_store_current_label_tl
4376   \tl_if_no_value:nF { #1 }
4377   {
4378     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
4379     {
4380       \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_opt_sep_tl }
4381       \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
4382     }
4383     \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
4384   }
4385   \__enumext_starred_item_viii_aux_ii:w
4386 }
4387 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
4388 {
4389   \legacy_if_set_true:n { @noitemarg }
4390   \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4391 }

```

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

`__enumext_starred_item_exec:`

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

```

4392 \cs_new_protected:Nn \__enumext_starred_item_exec:
4393 {
4394   \tl_put_left:Ne \l__enumext_store_current_label_tl { \l__enumext_label_viii_tl }

```

```

4395 \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
4396 \__enumext_keyans_store_ref:
4397 \tl_put_left:Ne \l__enumext_store_current_label_tl { \item }
4398 \__enumext_keyans_addto_seq_link:
4399 \int_gincr:N \g__enumext_check_starred_cmd_int
4400 \bool_if:NT \l__enumext_show_answer_bool
4401 {
4402   \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4403 }
4404 \bool_if:NT \l__enumext_show_position_bool
4405 {
4406   \tl_set:Ne \l__enumext_mark_answer_sym_tl
4407   {
4408     \group_begin:
4409     \exp_not:N \normalfont
4410     \exp_not:N \footnotesize [ \int_eval:n
4411       {
4412         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
4413       }
4414     ]
4415     \group_end:
4416   }
4417   \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4418 }
4419 }

```

(End of definition for `__enumext_starred_item_exec:`.)

12.44.2 Real definition of `\item` in keyans*

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

```

4420 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
4421 {
4422   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
4423   \legacy_if:nT { @noitemarg }
4424   {
4425     \legacy_if_set_false:n { @noitemarg }
4426     \legacy_if:nT { @nmbrlist }
4427     {
4428       \bool_if:NT \l__enumext_hyperref_bool
4429       {
4430         \legacy_if_set_true:n { @hyper@item }
4431       }
4432       \refstepcounter{enumXviii}
4433     }
4434   }

```

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

```

4435 \group_begin:
4436 \lrbox{ \l__enumext_item_text_viii_box }
4437 \bool_if:NF \l__enumext_footnotes_key_bool
4438 {
4439   \__enumext_renew_footnote:
4440 }
4441 \bool_if:NT \l__enumext_item_starred_viii_bool
4442 {
4443   \__enumext_starred_item_exec:
4444 }
4445 \group_begin:
4446 \tl_use:N \l__enumext_label_font_style_viii_tl
4447 \bool_if:NTF \l__enumext_wrap_label_viii_bool
4448 {
4449   \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
4450   { \__enumext_wrapper_label_viii:n [#1] }
4451 }
4452 {
4453   \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1
4454 }
4455 \group_end:
4456 \skip_horizontal:N \l__enumext_labelsep_viii_dim
4457 \tl_use:N \l__enumext_after_list_args_viii_tl
4458 \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }

```

```

4459         \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
4460         \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
4461         \bool_if:NT \l__enumext_item_starred_viii_bool
4462         {
4463             \tl_use:N \l__enumext_fake_item_indent_viii_tl
4464             \__enumext_keyans_show_item_opt:
4465             \skip_horizontal:n { -\l__enumext_fake_item_indent_viii_dim - \l__enumext_labelsep_viii_dim }
4466         }
4467         {
4468             \tl_use:N \l__enumext_fake_item_indent_viii_tl
4469         }
4470     }

```

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

`__enumext_stop_item_viii:` The function `__enumext_stop_item_viii:` shall terminate with the capture of `\item` and its *contents*. Close the environments `minipage`, `lrbox` and the group. Then we only have to set the width of the box and print it next to `\footnote`, and add the horizontal and vertical separation between the boxes.

```

4471 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
4472 {
4473     \__enumext_endminipage:
4474     \endlrbox
4475     \group_end:
4476     \box_set_wd:Nn \l__enumext_item_text_viii_box
4477     {
4478         \l__enumext_joined_width_viii_dim
4479         + \l__enumext_labelwidth_viii_dim
4480         + \l__enumext_labelsep_viii_dim
4481     }
4482     \int_set:Nn \hbadness { 10000 }
4483     \box_use_drop:N \l__enumext_item_text_viii_box
4484     \bool_if:NF \l__enumext_footnotes_key_bool
4485     {
4486         \__enumext_print_footnote:
4487     }
4488     \int_compare:nNnTF
4489     { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
4490     {
4491         \par\noindent
4492         \int_zero:N \l__enumext_item_column_pos_viii_int
4493     }
4494     { \hspace{ \l__enumext_columns_sep_viii_dim } }
4495 }

```

(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.

```

4496 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
4497 {
4498     \int_compare:nNnT
4499     {
4500         \int_mod:nn
4501         { \g__enumext_item_count_all_viii_int }
4502         { \l__enumext_columns_viii_int }
4503     }
4504     =
4505     { 0 }
4506     {
4507         \par
4508         \vspace{ -\l__enumext_itemsep_viii_skip }
4509         \int_gzero:N \g__enumext_item_count_all_viii_int
4510     }
4511 }

```

(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.

```

4512 \NewDocumentCommand \getkeyans { m }
4513 {
4514   \exp_args:Ne \__enumext_getkeyans_aux:n
4515   { \tl_to_str:e { \text_expand:n {#1} } }
4516 }

```

(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.

```

4517 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
4518 {
4519   \str_if_in:nnTF {#1} { : }
4520   {
4521     \use:e
4522     {
4523       \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
4524       { {##1} {##2} }
4525     }
4526     \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
4527   }
4528   { \msg_error:nnn { enumext } { missing-colon } {#1} }
4529 }

```

(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 $\langle prop\ list \rangle$.

```

4530 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
4531 {
4532   \prop_if_exist:cTF { g__enumext_#1_prop }
4533   {
4534     \prop_item:cn { g__enumext_#1_prop } {#2}
4535   }
4536   {
4537     \msg_error:nnn { enumext } { undefined-storage-anskey } {#1}
4538   }
4539 }

```

(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[$\langle print, level \rangle$]`.

```

4540 \keys_define:nn { enumext / print }
4541 {
4542   print* .code:n = \keys_precompile:neN { enumext / enumext* }
4543               { \__enumext_filter_save_key:n {#1} }
4544               \l__enumext_print_keyans_starred_tl, % starred cmd
4545   print* .initial:n = { nosep, label=\arabic*, columns=2, first=\small, font=\small },
4546   print-1 .code:n = \keys_precompile:neN { enumext / level-1 }
4547               { \__enumext_filter_save_key:n {#1} }
4548               \l__enumext_print_keyans_i_tl,
4549   print-1 .initial:n = { nosep, label=\arabic*, columns=2, first=\small, font=\small },
4550   print-2 .code:n = \keys_precompile:neN { enumext / level-2 }
4551               { \__enumext_filter_save_key:n {#1} }
4552               \l__enumext_print_keyans_ii_tl,
4553   print-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },

```

```

4554   print-3 .code:n      = \keys_precompile:neN { enumext / level-3 }
4555                       { \__enumext_filter_save_key:n {#1} }
4556                       \l__enumext_print_keyans_iii_tl,
4557   print-3 .initial:n   = { nosep, label=\roman*., first=\small, font=\small },
4558   print-4 .code:n      = \keys_precompile:neN { enumext / level-4 }
4559                       { \__enumext_filter_save_key:n {#1} }
4560                       \l__enumext_print_keyans_iv_tl,
4561   print-4 .initial:n   = { nosep, label=\Alph*., first=\small, font=\small },
4562   print-* .code:n      = \keys_precompile:neN { enumext / enumext* }
4563                       { \__enumext_filter_save_key:n {#1} }
4564                       \l__enumext_print_keyans_vii_tl, % starred nested
4565   print-* .initial:n   = { nosep, label=\arabic*., first=\small, font=\small },
4566   }

```

• 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`.

```

4567 \NewDocumentCommand \printkeyans { s O{} m }
4568 {
4569   \group_begin:
4570     \tl_use:N \l__enumext_print_keyans_i_tl
4571     \tl_use:N \l__enumext_print_keyans_ii_tl
4572     \tl_use:N \l__enumext_print_keyans_iii_tl
4573     \tl_use:N \l__enumext_print_keyans_iv_tl
4574     \tl_use:N \l__enumext_print_keyans_vii_tl
4575     \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
4576   \group_end:
4577 }

```

(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.

```

4578 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
4579 {
4580   \seq_if_exist:cTF { g__enumext_#3_seq }
4581   {
4582     \seq_if_empty:cF { g__enumext_#3_seq }
4583     {
4584       %%\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)*.

```

4585       \bool_if:nTF {#1}
4586       {
4587         \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
4588         {
4589           \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
4590         }
4591       }
4592       \tl_use:N \l__enumext_print_keyans_starred_tl
4593       \begin{enumext*}[#2]
4594         \keys_set:nn { enumext / level-1 }{ base-fix }
4595         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4596       \end{enumext*}
4597     }
4598   }

```

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)*.

```

4599   {
4600     \begin{enumext}[#2]
4601     \keys_set:nn { enumext / enumext* }{ base-fix }
4602     \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4603   \end{enumext}

```

```

4604         }
4605     }
4606 }
4607 {
4608     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
4609 }
4610 }

```

(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
4611 \cs_new:Npn \__enumext_filter_first_level:n #1
4612 {
4613     \use:e
4614     {
4615         \keyval_parse:NNn
4616         \__enumext_filter_first_level_key:n
4617         \__enumext_filter_first_level_pair:nn {#1}
4618     }
4619 }

```

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*`.

```

4620 \cs_new:Npn \__enumext_filter_first_level_key:n #1
4621 {
4622     \str_case:nnF {#1}
4623     {
4624         { resume } {}
4625         { resume* } {}
4626     }
4627     { , { \exp_not:n {#1} } } }
4628 }

```

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.

```

4629 \cs_new:Npn \__enumext_filter_first_level_pair:nn #1#2
4630 {
4631     \str_case:nnF {#1}
4632     {
4633         { series } {}
4634         { resume } {}
4635         { save-ans } {}
4636     }
4637     { , { \exp_not:n {#1} } = { \exp_not:n {#2} } } }
4638 }

```

(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`.

```

4639 \keys_define:nn { enumext / meta-families }
4640 {
4641     enumext-1 .code:n =
4642     {
4643         \keys_set:ne { enumext / level-1 }
4644         {
4645             \__enumext_filter_first_level:n {#1}
4646         }
4647     } ,
4648     enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
4649     enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
4650     enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
4651     keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} } ,
4652     enumext* .code:n =
4653     {
4654         \keys_set:ne { enumext / enumext* }

```



```

4655         {
4656             \__enumext_filter_first_level:n {#1}
4657         }
4658     } ,
4659     keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
4660     print* .code:n = { \keys_set:nn { enumext / print } { print* = {#1} } } ,
4661     print-1 .code:n = { \keys_set:nn { enumext / print } { print-1 = {#1} } } ,
4662     print-2 .code:n = { \keys_set:nn { enumext / print } { print-2 = {#1} } } ,
4663     print-3 .code:n = { \keys_set:nn { enumext / print } { print-3 = {#1} } } ,
4664     print-4 .code:n = { \keys_set:nn { enumext / print } { print-4 = {#1} } } ,
4665     print-* .code:n = { \keys_set:nn { enumext / print } { print-* = {#1} } } ,
4666     unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
4667 }

```

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

```

4668 \seq_const_from_clist:Nn \__enumext_all_families_seq
4669 {
4670     enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
4671     keyans*, print-1, print-2, print-3, print-4, print-*, print*,
4672 }

```

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

```

4673 \NewDocumentCommand \setenumext { 0{enumext,1} +m }
4674 {
4675     \seq_clear:N \__enumext_setkey_tmpa_seq
4676     \seq_set_from_clist:Nn \__enumext_setkey_tmpb_seq {#1}
4677     \int_set:Nn \__enumext_setkey_tmpa_int
4678     {
4679         \seq_count:N \__enumext_setkey_tmpb_seq
4680     }
4681     \int_compare:nNnTF { \__enumext_setkey_tmpa_int } > { 1 }
4682     {
4683         \seq_pop_left:NN \__enumext_setkey_tmpb_seq \__enumext_setkey_tmpa_tl
4684         \seq_map_function:NN \__enumext_setkey_tmpb_seq \__enumext_set_parse:n
4685         \seq_set_map:e:NNn \__enumext_setkey_tmpa_seq \__enumext_setkey_tmpa_seq
4686         {
4687             \tl_use:N \__enumext_setkey_tmpa_tl - ##1
4688         }
4689     }
4690     {
4691         \seq_put_right:Ne \__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
4692     }
4693     \seq_if_empty:NNTF \__enumext_setkey_tmpa_seq
4694     { \seq_map_inline:Nn \__enumext_all_families_seq }
4695     { \seq_map_inline:Nn \__enumext_setkey_tmpa_seq }
4696     {
4697         \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
4698     }
4699 }

```

(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.

```

4700 \cs_new_protected:Npn \__enumext_set_parse:n #1
4701 {
4702     \tl_set:Ne \__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
4703     \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
4704     { \tl_remove_all:Nn \__enumext_setkey_tmpb_tl {##1} }
4705     \tl_if_empty:NNTF \__enumext_setkey_tmpb_tl
4706     {
4707         \seq_put_right:Ne \__enumext_setkey_tmpa_seq
4708         { \tl_trim_spaces:n {#1} }
4709     }
4710     { \__enumext_set_error:nn {#1} { } }
4711 }
4712 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
4713 { \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
4714 \prop_const_from_keyval:Nn \c__enumext_meta_paths_prop
4715 {
4716   {enumext,1} = level-1,
4717   {enumext,2} = level-2,
4718   {enumext,3} = level-3,
4719   {enumext,4} = level-4,
4720   {enumext*} = enumext*
4721 }
```

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

```
4722 \NewDocumentCommand \setenumextmeta { s O{enumext,1} m +m }
4723 {
4724   \str_if_eq:eeTF { \tl_trim_spaces:n {#3} } { unknown }
4725   { \msg_error:nn { enumext } { prohibited-unknown } }
4726   {
4727     \bool_if:nTF {#1}
4728     {
4729       \int_step_inline:nn { 4 }
4730       { __enumext_add_meta_key:nnn { enumext, ##1 } {#3} {#4} }
4731       __enumext_add_meta_key:nnn { enumext* } {#3} {#4}
4732     }
4733     { __enumext_add_meta_key:nnn {#2} {#3} {#4} }
4734   }
4735 }
```

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

```
4736 \cs_new_protected:Npn __enumext_add_meta_key:nnn #1
4737 {
4738   \tl_set:Nn \l__enumext_meta_path_tl {#1}
4739   \tl_replace_all:Nnn \l__enumext_meta_path_tl { ~ } {}
4740   \prop_get:NVNTF
4741   \c__enumext_meta_paths_prop \l__enumext_meta_path_tl \l__enumext_meta_path_tl
4742   { __enumext_def_meta_key:Vnn \l__enumext_meta_path_tl }
4743   {
4744     \msg_error:nnn { enumext } { unknown-set } {#1}
4745     \use_none:nn
4746   }
4747 }
4748 \cs_new_protected:Npn __enumext_def_meta_key:nnn #1#2#3
4749 {
4750   \bool_lazy_or:nnTF
4751   { \keys_if_exist_p:nn { enumext / #1 } {#2} }
4752   { \keys_if_exist_p:nn { enumext / enumext* } {#2} }
4753   { \msg_error:nnn { enumext } { already-defined } {#2} }
4754   {
4755     \keys_define:nn { enumext / #1 }
4756     {
4757       #2 .meta:n = {#3},
4758       #2 .value_forbidden:n = true
4759     }
4760   }
4761 }
4762 \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:n
4763 \keys_define:nn { enumext / foreach }
```

©2024 by Pablo González L

```

4764 {
4765     before .tl_set:N = \l__enumext_foreach_before_tl,
4766     before .value_required:n = true,
4767     after  .tl_set:N = \l__enumext_foreach_after_tl,
4768     after  .value_required:n = true,
4769     start  .int_set:N = \l__enumext_foreach_start_int,
4770     start  .value_required:n = true,
4771     stop   .int_set:N = \l__enumext_foreach_stop_int,
4772     stop   .value_required:n = true,
4773     step   .int_set:N = \l__enumext_foreach_step_int,
4774     step   .value_required:n = true,
4775     wrapper .cs_set_protected:Np = \l__enumext_foreach_wrapper:n #1,
4776     wrapper .value_required:n = true,
4777     sep     .tl_set:N = \l__enumext_foreach_sep_tl,
4778     sep     .value_required:n = true,
4779     unknown .code:n = { \l__enumext_parse_foreach_keys:n {#1} }
4780 }
4781 \keys_precompile:nnN { enumext / foreach }
4782 {
4783     before={},after={},start=1,step=1,stop=0,wrapper=#1,sep=
4784 }
4785 \g__enumext_foreach_default_keys_tl

```

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

```

4786 \cs_new_protected:Npn \l__enumext_parse_foreach_keys:nn #1#2
4787 {
4788     \tl_if_blank:nTF {#2}
4789     {
4790         \msg_error:nnn { enumext } { for-key-unknown } {#1}
4791     }
4792     {
4793         \msg_error:nnnn { enumext } { for-key-value-unknown } {#1} {#2}
4794     }
4795 }
4796 \cs_new_protected:Npn \l__enumext_parse_foreach_keys:n #1
4797 {
4798     \exp_args:NV \l__enumext_parse_foreach_keys:nn \l_keys_key_str {#1}
4799 }

```

We create the command.

```

4800 \NewDocumentCommand \foreachkeyans { +0{ } m }
4801 {
4802     \l__enumext_foreach_keyans:nn {#1} {#2}
4803 }

```

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.

```

4804 \cs_new_protected:Npn \l__enumext_foreach_keyans:nn #1 #2
4805 {
4806     \tl_use:N \g__enumext_foreach_default_keys_tl
4807     \keys_set:nn { enumext / foreach } {#1}
4808     \tl_set:Nn \l__enumext_foreach_name_prop_tl {#2}
4809     \prop_if_exist:cF { g__enumext_#2_prop }
4810     {
4811         \msg_error:nnn { enumext } { undefined-storage-anskey } {#2}
4812     }
4813     \int_compare:nNnT { \l__enumext_foreach_stop_int } = { 0 }
4814     {
4815         \int_set:Nn \l__enumext_foreach_stop_int
4816         { \prop_count:c { g__enumext_#2_prop } }
4817     }
4818     \seq_clear:N \l__enumext_foreach_print_seq
4819     \int_step_function:nnnN
4820     { \l__enumext_foreach_start_int }
4821     { \l__enumext_foreach_step_int }
4822     { \l__enumext_foreach_stop_int }
4823     \l__enumext_foreach_add_body:n
4824     \seq_use:NV \l__enumext_foreach_print_seq \l__enumext_foreach_sep_tl
4825 }
4826 \cs_new_protected:Npn \l__enumext_foreach_add_body:n #1
4827 {
4828     \seq_put_right:Ne \l__enumext_foreach_print_seq

```

```

4829     {
4830         \exp_not:V \l__enumext_foreach_before_tl
4831         \__enumext_foreach_wrapper:n
4832         {
4833             \prop_item:cn { g__enumext_ \l__enumext_foreach_name_prop_tl _prop }{#1}
4834         }
4835         \exp_not:V \l__enumext_foreach_after_tl
4836     }
4837 }

```

(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.

```

4838 \msg_new:nnn { enumext } { package-load }
4839 {
4840     The ~ '#1' ~ package ~ is ~ already ~ loaded.
4841 }
4842 \msg_new:nnn { enumext } { package-not-load }
4843 {
4844     The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
4845 }
4846 \msg_new:nnn { enumext } { package-load-foot }
4847 {
4848     The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
4849 }

```

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

```

4850 \msg_new:nnn { enumext } { counters }
4851 {
4852     The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
4853     package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
4854 }

```

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

```

4855 \msg_new:nnn { enumext } { unknown-choice }
4856 {
4857     The ~ value ~ '#3' ~ for ~ '#1' ~ key ~ is ~ invalid ~ use ~ ('#2').
4858 }

```

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

```

4859 \msg_new:nnnn { enumext } { anskey-env-error }
4860 {
4861     The ~ '#1' ~ environment ~is ~ reserved ~ by ~\\
4862     'enumext' ~ package, ~ It~ is~ already~ defined.
4863 }
4864 {
4865     The ~ anskey* ~ environment ~ is ~ defined ~ internally ~
4866     for ~ the ~ 'save-ans' ~ key.\\
4867 }

```

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

```

4868 \msg_new:nnn { enumext } { store-prop }
4869 {
4870     * ~ Package ~ enumext: ~ Creating ~
4871     \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
4872 }
4873 \msg_new:nnn { enumext } { store-seq }
4874 {
4875     * ~ Package ~ enumext: ~ Creating ~
4876     \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
4877 }
4878 \msg_new:nnn { enumext } { store-int }
4879 {
4880     * ~ Package ~ enumext: ~ Creating ~
4881     \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_context:.
4882 }
4883 \msg_new:nnn { enumext } { prop-seq-int-hook }
4884 {
4885     * ~ Package ~ enumext: ~ Elements ~ in ~
4886     \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
4887     * ~ Package ~ enumext: ~ Elements ~ in ~

```

```

4888     \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
4889     * ~ Package ~ enumext: ~ Value ~ off ~
4890     \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
4891 }
4892 \msg_new:nnn { enumext } { item-answer-hook }
4893 {
4894     * ~ Package ~ enumext: ~ Value ~ off ~
4895     \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\\
4896     * ~ Package ~ enumext: ~ Value ~ off ~
4897     \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\\
4898     * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
4899 }

```

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

```

4900 \msg_new:nnn { enumext } { invalid-key }
4901 {
4902     The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
4903 }
4904 \msg_new:nnn { enumext } { unknown-key-family }
4905 {
4906     Unknown~key~family~`\l_keys_key_str'~for~enumext.
4907 }

```

Messages used in length calculation.

```

4908 \msg_new:nnn { enumext } { width-negative }
4909 {
4910     Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
4911     The ~ key ~ '#1'~ accepts ~ values ~ >= ~ 0pt.
4912 }
4913 \msg_new:nnn { enumext } { width-zero }
4914 {
4915     Invalid ~ '#1=#2' ~ \msg_line_context:.\
4916     The ~ key ~ '#1'~ accepts ~ values ~ > ~ 0pt.
4917 }

```

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

```

4918 \msg_new:nnn { enumext } { list-lengths }
4919 {
4920     **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
4921     \__enumext_show_length:nnn { dim } { labelsep } { #1}
4922     \__enumext_show_length:nnn { dim } { labelwidth } { #1}
4923     \__enumext_show_length:nnn { dim } { itemindent } { #1}
4924     \__enumext_show_length:nnn { dim } { leftmargin } { #1}
4925     \__enumext_show_length:nnn { dim } { rightmargin } { #1}
4926     \__enumext_show_length:nnn { dim } { listparindent } { #1}
4927     \__enumext_show_length:nnn { skip } { topsep } { #1}
4928     \__enumext_show_length:nnn { skip } { parsep } { #1}
4929     \__enumext_show_length:nnn { skip } { partopsep } { #1}
4930     \__enumext_show_length:nnn { skip } { itemsep } { #1}
4931     ****
4932 }

```

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

```

4933 \msg_new:nnn { enumext } { list-lengths-not-nested }
4934 {
4935     **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
4936     \__enumext_show_length:nnn { dim } { labelsep } { #1}
4937     \__enumext_show_length:nnn { dim } { labelwidth } { #1}
4938     \__enumext_show_length:nnn { dim } { itemindent } { #1}
4939     \__enumext_show_length:nnn { dim } { leftmargin } { #1}
4940     \__enumext_show_length:nnn { dim } { rightmargin } { #1}
4941     \__enumext_show_length:nnn { dim } { listparindent } { #1}
4942     \__enumext_show_length:nnn { skip } { topsep } { #1}
4943     \__enumext_show_length:nnn { skip } { parsep } { #1}
4944     \__enumext_show_length:nnn { skip } { partopsep } { #1}
4945     \__enumext_show_length:nnn { skip } { itemsep } { #1}
4946     ****
4947 }

```

Messages used by `ref` key.

```

4948 \msg_new:nnn { enumext } { key-ref-empty }
4949 {
4950     Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4951 }

```

Messages used by `save-ans` key.

```

4952 \msg_new:nnn { enumext } { save-ans-empty }
4953 {
4954   Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1' ~ \msg_line_context:.
4955 }
4956 \msg_new:nnn { enumext } { save-ans-log }
4957 {
4958   * ~ Package ~ enumext: ~ Start ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
4959 }
4960 \msg_new:nnn { enumext } { save-ans-log-hook }
4961 {
4962   * ~ Package ~ enumext: ~ Stop ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
4963 }
4964 \msg_new:nnn { enumext } { save-ans-hook }
4965 {
4966   Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
4967 }

```

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

```

4968 \msg_new:nnn { enumext } { need-save-ans }
4969 {
4970   Key ~ '#1' ~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2' ~ \msg_line_context:.
4971 }
4972 \msg_new:nnn { enumext } { items-same-answer }
4973 {
4974   *****\
4975   * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~
4976   for ~ \c_left_brace_str #2 \c_right_brace_str\
4977   * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~
4978   'OK', ~ all ~ items ~ with ~ answer.\
4979   *****
4980 }
4981 \msg_new:nnn { enumext } { item-greater-answer }
4982 {
4983   Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\
4984   started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\
4985   Items ~ > ~ Answers.
4986 }
4987 \msg_new:nnn { enumext } { item-less-answer }
4988 {
4989   Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\
4990   started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\
4991   Items ~ < ~ Answers.
4992 }

```

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

```

4993 \msg_new:nnn { enumext } { missing-starred }
4994 {
4995   Missing ~ '\c_backslash_str #1*' ~ #2.
4996 }
4997 \msg_new:nnn { enumext } { many-starred }
4998 {
4999   Many ~ '\c_backslash_str #1*' ~ #2.
5000 }

```

Messages used by `\printkeyans*` command.

```

5001 \msg_new:nnn { enumext } { print-starred }
5002 {
5003   \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
5004   #2 ~ environment ~ \msg_line_context:.
5005 }

```

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

```

5006 \msg_new:nnn { enumext } { list-too-deep }
5007 {
5008   Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \
5009   The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
5010 }

```

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

```

5011 \msg_new:nnn { enumext } { anskey-unnumber-item }
5012 {
5013   Can't ~ store ~ with ~ a ~ unnumbered ~ \c_backslash_str item ~ \msg_line_context:.

```

```

5014     }
5015     \msg_new:nnn { enumext } { anskey-already-stored }
5016     {
5017         Content ~ already ~ stored ~ for ~ this ~ \c_backslash_str item ~ \msg_line_context:.
5018     }
5019     \msg_new:nnn { enumext } { anskey-empty-arg }
5020     {
5021         Can't ~ store ~ empty ~ content ~ \msg_line_context:.
5022     }
5023     \msg_new:nnn { enumext } { anskey-wrong-place }
5024     {
5025         Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5026         '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5027     }
5028     \msg_new:nnn { enumext } { anskey-nested }
5029     {
5030         The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
5031     }
5032     \msg_new:nnn { enumext } { anskey-math-mode }
5033     {
5034         #1 ~ can't ~ work ~ in ~ math ~ mode ~ \msg_line_context:.
5035     }
5036     \msg_new:nnn { enumext } { anskey-env-wrong }
5037     {
5038         The ~ environment ~ anskey* ~ cannot ~ use ~ in ~ '#1' ~ \msg_line_context:.
5039     }
5040     \msg_new:nnn { enumext } { ansPIC-wrong-place }
5041     {
5042         Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5043         '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5044     }
5045     \msg_new:nnn { enumext } { command-wrong-place }
5046     {
5047         Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5048         '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
5049     }
5050     \msg_new:nnnn { enumext } { anskey-env-key-unknown }
5051     {
5052         The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5053         'anskey*' ~ and ~ is ~ being ~ ignored.
5054     }
5055     {
5056         The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5057         Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5058     }
5059     \msg_new:nnnn { enumext } { anskey-env-key-value-unknown }
5060     {
5061         The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5062         'anskey*' ~ and ~ is ~ being ~ ignored.
5063     }
5064     {
5065         The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5066         Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5067     }
5068     \msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
5069     { The ~ key ~'#1'~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.}
5070     {
5071         The ~ command ~'\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5072         Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5073     }
5074     \msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
5075     { The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.}
5076     {
5077         The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5078         Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5079     }

```

Messages used by [keyans](#), [keyans*](#) and [keyansPIC](#) environment.

```

5080     \msg_new:nnn { enumext } { keyans-nested }
5081     {
5082         The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
5083     }

```



```

5084 \msg_new:nnn { enumext } { keyans-wrong-level }
5085 {
5086   Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
5087   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5088 }
5089 \msg_new:nnn { enumext } { wrong-place }
5090 {
5091   Wrong ~ place ~ for ~ '#1' ~ environment ~ \msg_line_context:~ \\
5092   '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext'.
5093 }
5094 \msg_new:nnn { enumext } { keyanspic-nested }
5095 {
5096   The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested~ \msg_line_context:~.
5097 }
5098 \msg_new:nnn { enumext } { keyanspic-wrong-level }
5099 {
5100   Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:~ \\
5101   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5102 }
5103 \msg_new:nnn { enumext } { keyanspic-item-cmd }
5104 {
5105   Can't ~ use ~ \c_backslash_str item ~ in ~ keyanspic ~ \msg_line_context:.
5106 }
5107 \msg_new:nnnn { enumext } { keyans-unknown-key }
5108 {
5109   The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5110   '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5111 }
5112 {
5113   The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5114   ~ have ~ a ~ key ~ called ~ '#1'.\\
5115   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5116 }
5117 \msg_new:nnnn { enumext } { keyans-unknown-key-value }
5118 {
5119   The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5120   '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5121 }
5122 {
5123   The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5124   ~ have ~ a ~ key ~ called ~ '#1'.\\
5125   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5126 }

```

Message used by unknown $\langle keys \rangle$ in `enumext*`. environment.

```

5127 \msg_new:nnnn { enumext } { starred-unknown-key }
5128 {
5129   The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5130   '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5131 }
5132 {
5133   The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5134   ~ have ~ a ~ key ~ called ~ '#1'.\\
5135   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5136 }
5137 \msg_new:nnnn { enumext } { starred-unknown-key-value }
5138 {
5139   The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5140   '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5141 }
5142 {
5143   The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5144   ~ have ~ a ~ key ~ called ~ '#1'.\\
5145   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5146 }

```

Message used by unknown $\langle keys \rangle$ in `enumext` environment.

```

5147 \msg_new:nnnn { enumext } { standar-unknown-key }
5148 {
5149   The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_tl
5150   ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5151 }

```

```

5152 {
5153     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5154     ~ have ~ a ~ key ~ called ~ '#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5155     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5156 }
5157 \msg_new:nnnn { enumext } { standar-unknown-key-value }
5158 {
5159     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_
5160     ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5161 }
5162 {
5163     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5164     ~ have ~ a ~ key ~ called ~ '#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5165     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5166 }

```

Message used by unknown *(keys)* in `\foreachkeyans`.

```

5167 \msg_new:nnnn { enumext } { for-key-unknown }
5168 { The~key~'#1'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored.}
5169 {
5170     The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5171     Check~that~you~have~spelled~the~key~name~correctly.
5172 }
5173 \msg_new:nnnn { enumext } { for-key-value-unknown }
5174 { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored. }
5175 {
5176     The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5177     Check~that~you~have~spelled~the~key~name~correctly.
5178 }

```

Messages used by `\getkeyans` command.

```

5179 \msg_new:nnn { enumext } { undefined-storage-anskey }
5180 {
5181     Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
5182 }

```

Messages used by `\miniright` command.

```

5183 \msg_new:nnn { enumext } { missing-miniright }
5184 {
5185     Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\\
5186     The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
5187 }
5188 \msg_new:nnn { enumext } { wrong-miniright-place }
5189 {
5190     Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
5191     Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
5192 }
5193 \msg_new:nnn { enumext } { wrong-miniright-use }
5194 {
5195     Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
5196     '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
5197 }
5198 \msg_new:nnn { enumext } { wrong-miniright-starred }
5199 {
5200     Can't ~ use ~ \c_backslash_str miniright ~ in ~ starred ~ environments ~ \msg_line_context:.
5201 }
5202 \msg_new:nnn { enumext } { many-miniright-used }
5203 {
5204     Can't ~ use ~ \c_backslash_str miniright ~ more ~ than ~ once ~ \msg_line_context:.
5205 }

```

Messages used by `\setenumextmeta` command.

```

5206 \msg_new:nnn { enumext } { unknown-set }
5207 {
5208     Argument ~ [#1] ~ is ~ unknown ~ by ~ \c_backslash_str setenumextmeta ~ \msg_line_context:.
5209 }
5210 \msg_new:nnn { enumext } { already-defined }
5211 {
5212     The ~ key ~ '#1' ~ is ~ already ~ defined ~ \msg_line_context:.
5213 }
5214 \msg_new:nnn { enumext } { prohibited-unknown }
5215 {
5216     The ~ name ~ 'unknown' ~ can't ~ be ~ chosen~ for ~ a ~ meta ~ key ~ \msg_line_context:.

```

```
5217 }
```

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

```
5218 \msg_new:nnn { enumext } { nested }
```

```
5219 {
```

```
5220   The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ \msg_line_con
```

```
5221 }
```

```
5222 \msg_new:nnn { enumext } { nested-horizontal }
```

```
5223 {
```

```
5224   The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ in ~ '#1' ~ \
```

```
5225 }
```

```
5226 \msg_new:nnn { enumext } { item-joined }
```

```
5227 {
```

```
5228   Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~\msg_line_context:.
```

```
5229 }
```

```
5230 \msg_new:nnn { enumext } { item-joined-columns }
```

```
5231 {
```

```
5232   Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.
```

```
5233 }
```

12.51 Finish package

Finish package implementation.

```
5234 \file_input_stop:
```

```
5235 \</package>
```

13 Index of Implementation

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

Symbols	
<code>*</code>	219
<code>\+</code>	211
<code>\-</code>	211
<code>\\</code>	227, 2612, 3615, 4852, 4861, 4866, 4886, 4888, 4895, 4897, 4910, 4915, 4920, 4935, 4974, 4976, 4978, 4983, 4984, 4989, 4990, 5008, 5025, 5042, 5047, 5056, 5065, 5071, 5077, 5086, 5091, 5100, 5114, 5124, 5134, 5144, 5154, 5164, 5170, 5176, 5185, 5190, 5195
A	
<code>above</code>	<u>1427</u>
<code>above*</code>	<u>1427</u>
<code>\addvspace</code>	1121, 1149, 1183, 1286, 1349, 1355, 1392, 1414, 3424, 3440, 3444, 3445, 3446, 3569, 3584, 3942, 3956, 3997, 4011
<code>after</code>	<u>960</u>
<code>align</code>	<u>509</u>
<code>\Alph</code>	36, <u>41</u>
<code>\Alph</code>	461, 576, 621, 689, 4561
<code>\alph</code>	36, <u>41</u>
<code>\alph</code>	462, 574, 4553
<code>\anskey</code>	12, 73, 74, <u>2430</u>
<code>anskey*</code>	13, <u>2540</u>
<code>\anspic</code>	15, 98, <u>3593</u>
<code>\anspic*</code>	66
<code>\arabic</code>	30, <u>36</u>
<code>\arabic</code>	460, 573, 620, 4545, 4549, 4565
B	
<code>base-fix</code>	<u>819</u>
<code>\baselineskip</code>	<u>50</u>
<code>\baselineskip</code>	836, 847
<code>before</code>	<u>960</u>
<code>before*</code>	<u>960</u>
<code>below</code>	<u>1427</u>
<code>below*</code>	<u>1427</u>
bool commands:	
<code>\bool_gset_false:N</code>	332, 333, 334, 2716, 2718, 3958, 3962, 4013
<code>\bool_gset_true:N</code>	240, 250, 1063, 1920, 1926, 3934, 3959, 3989, 4014
<code>\bool_if:NTF</code>	400, 412, 429, 1371, 1449, 1463, 1476, 1487, 1498, 1509, 1520, 1531, 1580, 1597, 1602, 1610, 1637, 1675, 1680, 1687, 1691, 1713, 1718, 1726, 1733, 1764, 1772, 1865, 2063, 2073, 2152, 2176, 2183, 2207, 2305, 2327, 2367, 2380, 2384, 2434, 2453, 2477, 2531, 2542, 2631, 2668, 2732, 2765, 2780, 2855, 2866, 2870, 2889, 2902, 2944, 2978, 3013, 3147, 3209, 3219, 3251, 3256, 3359, 3407, 3422, 3430, 3497, 3554, 3567, 3575, 3595, 3930, 3939, 3943, 3985, 3994, 3998, 4087, 4097, 4180, 4185, 4194, 4198, 4213, 4242, 4298, 4400, 4404, 4428, 4437, 4441, 4447, 4461, 4484
<code>\bool_if:nTF</code>	1393, 1415, 3000, 3129, 3167, 3616, 4585, <u>4727</u>
<code>\bool_if_p:N</code>	259, 274, 832, 833, 843, 844, 1744, 1745, 1753, 1754, 1878, 1904, 1917, 1918, 1923, 1924, 2240, 2250, 2262, 2277, 2278, 2312, 2353, 2354, 2654, 2842, 2843, 2880, 2881, 3332, 3334, 3345, 3623, 3624
<code>\bool_lazy_all:nTF</code>	257, 272, 1876, 1902, 2238, 2247, 2260, 2275, 3330, 3343
<code>\bool_lazy_and:nnTF</code>	236, 246, 831, 842, 1364, 1743, 1752, 1916, 1922, 2311, 2318, 2352, 2495, 2507, 2653, 2659, 2841
<code>\bool_lazy_or:nnTF</code>	1805, 1812, 2879, 3622, 4750
<code>\bool_new:N</code>	34, 35, 36, 37, 38, 39, 40, 41, 64, 73, 95, 100, 101, 106, 107, 110, 135, 136, 144, 145, 150, 152, 153, 167, 179, 181
<code>\bool_not_p:n</code>	237, 247, 2249, 2313, 2319, 2655, 2660, 3333, 3346
<code>\bool_set_eq:NN</code>	2953, 3109, 4130, 4354
<code>\bool_set_false:N</code>	409, 853, 1850, 1851, 1883, 1888, 1892, 1896, 1909, 2595, 3307, 3455, 3505, 3589, 3654, 3672, 4055, 4082, 4127, 4304, 4351
<code>\bool_set_true:N</code>	264, 265, 279, 280, 391, 395, 502, 868, 1433, 1438, 1700, 1822, 1823, 2095, 2103, 2596, 2947, 2949, 2981, 2983, 3105, 3117, 3231, 3306, 3339, 3352, 3378, 3502, 3529, 3919, 3974, 4054, 4136, 4143, 4144, 4188, 4302, 4360, 4367, 4368
box commands:	
<code>\box_dp:N</code>	1192, 1198, 1204, 1218, 1219, 1220, 1223, 1233, 1237, 1246, 1253, 1258, 1266, 1295, 1296, 1299, 1306, 1319, 1327, 1333, 1341, 3446, 3684
<code>\box_ht:N</code>	1183
<code>\box_new:N</code>	70, 174, 180
<code>\box_set_wd:Nn</code>	4234, 4476
<code>\box_use_drop:N</code>	3954, 4009, 4241, 4483
<code>\box_wd:N</code>	468
C	
<code>\c</code>	219, 220, 726, 728, 740, 742
<code>\catcode</code>	2612
<code>\cB</code>	220
<code>\cE</code>	220
<code>\centering</code>	1395, 1417, 3710, 3947, 4002
<code>check-ans</code>	<u>1842</u>
Document class:	
<code>article</code>	42
clist commands:	
<code>\clist_const:Nn</code>	186
<code>\clist_map_function:nN</code>	3697
<code>\clist_map_inline:Nn</code>	508, 774, 959, 974, 1055, 1443
<code>\clist_map_inline:nn</code>	49, 60, 78, 85, 97, 109, 138, 161, 185, 536, 556, 828, 873, 894, 1069, 1549, 1789, 1856, 2042, 2060, 2092, 2235, 2774, 3034, 3046, 3086, 3196, 3199, 3226, 3238, 3241, 3261, 4703
<code>\columnbreak</code>	73
<code>\columnbreak</code>	2315
<code>columns</code>	<u>1039</u>
<code>columns-sep</code>	<u>1039</u>
<code>\columnsep</code>	94
<code>\columnsep</code>	3402, 3551
<code>\columnseprule</code>	94
<code>\columnseprule</code>	3405, 3553
Commands provide by enumext :	
<code>\anskey</code>	28, 63, 64, 68–72, 74, 75, 82, 83, 93, 108, 117, 118, 125
<code>\anspic*</code>	28, 29, 66, 69, 81–83, 98, 100, 101, 117, 118
<code>\anspic</code>	70, 98, 100, 125

131 / 144

```

\__enumext_after_stop_list: ... 46, 96, 975, 983,
    3452
\__enumext_after_stop_list_v: . 991, 999, 3590
\l__enumext_after_stop_list_v_tl . . . . . 1001
\__enumext_after_stop_list_vii: . . 108, 1007,
    1023, 4079
\l__enumext_after_stop_list_vii_tl . . . 1025
\__enumext_after_stop_list_viii: . 1027, 4326
\l__enumext_after_stop_list_viii_tl . . . 1029
\l__enumext_align_label_vii_str . . 4215, 4219
\l__enumext_align_label_viii_str . 4449, 4453
\l__enumext_align_label_X_str . . . . . 165
\c__enumext_all_envs_clist . . 186, 508, 774, 959,
    974, 1055, 1443
\c__enumext_all_families_seq . . 120, 4668, 4694
\l__enumext_anskey_env_bool 31, 77, 34, 265, 280,
    2542
\__enumext_anskey_env_clean_vars: . 80, 2647,
    2651, 2714
\__enumext_anskey_env_define_keys: 78, 2540,
    2549, 2620
\__enumext_anskey_env_exec: 79, 2545, 2616, 2616
\__enumext_anskey_env_make:n 63, 77, 1825, 2540,
    2540, 2548
\__enumext_anskey_env_reset_keys: 78, 79, 2585,
    2648
\__enumext_anskey_env_reset_keys:\__-
    enumext_rescan_anskey_env:n . . . . . 2540
\__enumext_anskey_env_save_keys: . . 79, 2628,
    2651, 2651
\__enumext_anskey_env_store: . . 80, 2644, 2651,
    2693
\__enumext_anskey_env_unknown:n 78, 2568, 2571
\__enumext_anskey_env_unknown:nn . 2573, 2575
\l__enumext_anskey_level_int . . 28, 2472, 2473
\__enumext_anskey_safe_inner: . 76, 2445, 2451,
    2470
\__enumext_anskey_safe_inner:n . . . . . 75
\__enumext_anskey_safe_outer: . 75, 2432, 2451,
    2451
\__enumext_anskey_show_wrap_arg:n . 74, 2364,
    2364, 2382, 2397
\__enumext_anskey_show_wrap_left:n 74, 2309,
    2378, 2378
\__enumext_anskey_unknown:n 75, 2400, 2414, 2416
\__enumext_anskey_unknown:nn . 2400, 2418, 2420
\__enumext_anskey_wrapper:n . . . . . 2005, 2376
\__enumext_at_begin_document:n . . 33, 193, 193,
    357, 363
\l__enumext_base_line_fix_bool . 823, 833, 844,
    853
\__enumext_before_args_exec: . . 46, 94, 107, 975,
    975, 3367
\__enumext_before_args_exec_v: 991, 991, 3522
\__enumext_before_args_exec_vii: . 1007, 1007,
    4073
\__enumext_before_args_exec_viii: 1011, 4320
\__enumext_before_env:nn 78, 197, 201, 2493, 2505,
    2517, 2618
\__enumext_before_keys_exec: 46, 975, 979, 3272
\__enumext_before_keys_exec_v: 991, 995, 3468
\__enumext_before_keys_exec_vii . . . . . 1007
\__enumext_before_keys_exec_vii: . 1015, 4027
\__enumext_before_keys_exec_viii: 1019, 4276

\__enumext_before_list: ... 94, 3266, 3364, 3364
\__enumext_before_list_v: ... 3463, 3519, 3519
\__enumext_before_list_vii: ... 107, 4022, 4069,
    4069
\__enumext_before_list_viii: . . 112, 4272, 4317,
    4317
\l__enumext_before_no_starred_key_v_tl 997
\l__enumext_before_no_starred_key_vii_-
    tl . . . . . 1017
\l__enumext_before_no_starred_key_viii_-
    tl . . . . . 1021
\l__enumext_before_starred_key_v_tl . . . 993
\l__enumext_before_starred_key_vii_tl . 1009
\l__enumext_before_starred_key_viii_tl 1013
\__enumext_calc_hspace:NNNNNNN 90, 3155, 3155,
    3186, 3191, 3233
\__enumext_check_ans_active: . 64, 94, 107, 1861,
    1861, 3368, 4072
\g__enumext_check_ans_item_tl . . . . . 83
\g__enumext_check_ans_key_bool 65, 66, 144, 332,
    1920, 1926, 2732
\l__enumext_check_ans_key_bool 65, 1846, 1851,
    1917, 1923
\__enumext_check_ans_key_hook: 65, 96, 108, 1914,
    1914, 3453, 4080
\__enumext_check_ans_level: 64, 1861, 1867, 1871
\__enumext_check_ans_log: 65, 66, 81, 1960, 1960,
    2736
\__enumext_check_ans_log_msg_greater: 1960,
    1966, 1979
\__enumext_check_ans_log_msg_less: 1960, 1964,
    1969
\__enumext_check_ans_log_msg_same_ok: 1960,
    1965, 1974
\__enumext_check_ans_msg_greater: 1936, 1942,
    1955
\__enumext_check_ans_msg_less: 1936, 1940, 1945
\__enumext_check_ans_msg_same_ok: 1936, 1941,
    1950
\__enumext_check_ans_show: . . 65, 81, 1936, 1936,
    2734
\l__enumext_check_answers_bool . 63, 64, 75, 85,
    144, 1823, 1850, 1865, 2152, 2176, 2183, 2207, 2434,
    2631, 2855, 2944, 2978, 4185
\__enumext_check_starred_cmd:n 32, 66, 83, 1984,
    1984, 3474, 3651, 4286
\g__enumext_check_starred_cmd_int 144, 1987,
    1993, 1998, 3123, 3621, 4399
\l__enumext_check_start_line_env_tl . 32, 144,
    295, 303, 311, 1990, 1996, 1999
\l__enumext_columns_sep_v_dim 3541, 3543, 3551
\l__enumext_columns_sep_vii_dim . . 3751, 3753,
    3762, 3774, 3850, 4251
\l__enumext_columns_sep_viii_dim . 3782, 3784,
    3793, 3805, 3899, 4494
\l__enumext_columns_v_int 1214, 3539, 3547, 3559,
    3564
\l__enumext_columns_vii_int . . 3756, 3759, 3763,
    3772, 3814, 3818, 3821, 3827, 3833, 3837, 4246, 4257
\l__enumext_columns_viii_int . 3787, 3790, 3794,
    3803, 3863, 3867, 3870, 3876, 3882, 3886, 4489, 4502
\l__enumext_counter_i_tl . . . . . 45, 447
\l__enumext_counter_ii_tl . . . . . 45, 448
\l__enumext_counter_iii_tl . . . . . 45, 449
\l__enumext_counter_iv_tl . . . . . 45, 450

```


`\c__enumext_counter_style_tl` 30, 50, 217
`\g__enumext_counter_styles_tl` . 26, 36, 67, 458, 476
`\l__enumext_counter_v_tl` 45, 451, 702
`\l__enumext_counter_vi_tl` 45, 452
`\l__enumext_counter_vii_tl` 45, 453, 632
`\l__enumext_counter_viii_tl` 45, 454, 649
`\l__enumext_current_widest_dim` 26, 67, 482, 565, 612, 683, 687
`__enumext_def_meta_key:nnn` . . . 121, 4714, 4742, 4748, 4762
`__enumext_default_item:n` . . . 2940, 2940, 3004
`__enumext_define_counters:Nn` 26, 438, 438, 447, 448, 449, 450, 451, 452, 453, 454
`__enumext_endminipage:` . 33, 363, 366, 380, 3712, 4231, 4473
`\g__enumext_envir_name_tl` 31, 34, 266, 281, 340, 1793, 1798, 1808, 1948, 1953, 1958, 1972, 1977, 1982
`\l__enumext_envir_name_tl` . 31, 32, 34, 235, 245, 294, 302, 310, 5110, 5113, 5120, 5123, 5130, 5133, 5140, 5143, 5149, 5153, 5159, 5163, 5220, 5224
`__enumext_execute_after_env:` 32, 33, 62, 65, 66, 76, 81, 2722, 2722, 3458, 4267
`__enumext_fake_item:` 899, 899, 3218
`\l__enumext_fake_item_indent_v_dim` 918, 923
`\l__enumext_fake_item_indent_v_tl` 920, 3106, 3110, 3118
`\l__enumext_fake_item_indent_vii_dim` 930, 935
`\l__enumext_fake_item_indent_vii_tl` 932, 4227
`\l__enumext_fake_item_indent_viii_dim` . 942, 947, 4465
`\l__enumext_fake_item_indent_viii_tl` . . 944, 4463, 4468
`\l__enumext_fake_item_indent_X_tl` 98
`__enumext_fake_item_vii:` 899, 927, 3250
`__enumext_fake_item_viii:` 899, 939, 3255
`__enumext_filter_first_level:n` . . 119, 4611, 4611, 4645, 4656
`__enumext_filter_first_level_key:n` 119, 4611, 4616, 4620
`__enumext_filter_first_level_pair:nn` . 119, 4611, 4617, 4629
`__enumext_filter_save_key:n` . . 69, 2068, 2076, 2099, 2105, 2107, 2107, 4543, 4547, 4551, 4555, 4559, 4563
`__enumext_filter_save_key_key:n` . . 69, 2107, 2112, 2116
`__enumext_filter_save_key_pair:nn` 69, 2107, 2113, 2124
`__enumext_filter_series:n` 57, 1550, 1550, 1588, 1600, 1605
`__enumext_filter_series_key:n` 58, 1550, 1555, 1559
`__enumext_filter_series_pair:nn` . . 58, 1550, 1556, 1567
`\g__enumext_footnote_arg_seq` . 162, 3722, 3735, 3745
`\g__enumext_footnote_int` . 162, 3729, 3732, 3734, 3736
`\g__enumext_footnote_int_seq` . 162, 3723, 3736, 3741, 3744
`__enumext_footnotes_key_bool` 34
`\l__enumext_footnotes_key_bool` 29, 34, 110, 152, 395, 400, 409, 4194, 4242, 4437, 4484
`__enumext_footnotetext:nn` . . . 3716, 3716, 3746
`__enumext_foreach_add_body:n` . 122, 4763, 4823, 4826
`\l__enumext_foreach_after_tl` 4767, 4835
`\l__enumext_foreach_before_tl` 4765, 4830
`\g__enumext_foreach_default_keys_tl` 121, 124, 4785, 4806
`__enumext_foreach_keyans:nn` . . 122, 4763, 4802, 4804
`\l__enumext_foreach_name_prop_tl` . 124, 4808, 4833
`\l__enumext_foreach_print_seq` 124, 4818, 4824, 4828
`\l__enumext_foreach_sep_tl` 4777, 4824
`\l__enumext_foreach_start_int` 4769, 4820
`\l__enumext_foreach_step_int` 4773, 4821
`\l__enumext_foreach_stop_int` . 4771, 4813, 4815, 4822
`__enumext_foreach_wrapper:n` 4775, 4831
`__enumext_getkeyans:nn` . . 117, 4526, 4530, 4530
`__enumext_getkeyans_aux:n` 117, 4514, 4517, 4517
`\l__enumext_hyperref_bool` . 29, 34, 35, 152, 391, 412, 429, 2354, 2843, 4180, 4428
`__enumext_hypertarget:nn` 35, 386, 414, 418, 434
`__enumext_if_is_int:n` 209
`__enumext_if_is_int:nTF` 209, 721, 735
`__enumext_internal_mini_page:` 34, 92, 107, 368, 368, 3301, 4043
`__enumext_is_not_nested:` 26, 31, 92, 107, 229, 229, 3302, 4044
`__enumext_is_on_first_level:` . 26, 31, 93, 107, 229, 255, 3308, 4056
`\g__enumext_item_anskey_int` 75, 83, 144, 327, 354, 355, 1933, 2303, 2857
`__enumext_item_answer_diff:` . . 65, 66, 81, 1929, 1929, 2729
`\g__enumext_item_answer_diff_int` . 65, 66, 144, 328, 1931, 1938, 1962
`\l__enumext_item_column_pos_vii_int` 108, 3821, 3827, 3833, 3837, 3844, 4108, 4246, 4249
`\l__enumext_item_column_pos_viii_int` . . 113, 3870, 3876, 3882, 3886, 3893, 4332, 4489, 4492
`\l__enumext_item_column_pos_X_int` 165
`\g__enumext_item_count_all_vii_int` 108, 3845, 4109, 4257, 4264
`\g__enumext_item_count_all_viii_int` 113, 3894, 4333, 4501, 4509
`\g__enumext_item_count_all_X_int` 165
`\g__enumext_item_number_bool` 144
`\l__enumext_item_number_bool` 64, 150, 1883, 1888, 1892, 1896, 1909, 2477, 2531, 2947, 2981, 4188
`\g__enumext_item_number_int` 64, 65, 144, 326, 353, 355, 1882, 1887, 1891, 1895, 1908, 1933, 2946, 2980, 4187
`__enumext_item_peek_args_vii:` 108, 4110, 4112, 4112
`__enumext_item_peek_args_viii:` . . 113, 4334, 4336, 4336
`__enumext_item_star_exec:` . 86, 2959, 2986, 3015
`\l__enumext_item_starred_vii_bool` 4127, 4143, 4198
`\l__enumext_item_starred_viii_bool` 4351, 4367, 4441, 4461
`\l__enumext_item_starred_X_bool` 165

__enumext_item_std:w . . 33, 85, 88, 100, 357, 361,
 2950, 2956, 2984, 3106, 3110, 3118, 3688
 \g__enumext_item_symbol_aux_tl . 85, 128, 2964,
 2967, 2992, 3020
 \g__enumext_item_symbol_aux_vii_tl 4151, 4200,
 4203, 4207, 4209
 \g__enumext_item_symbol_aux_X_tl 165
 \l__enumext_item_symbol_sep_vii_dim . . 4160,
 4168, 4206, 4208
 \l__enumext_item_symbol_vii_tl 4203
 \l__enumext_item_text_vii_box 4193, 4234, 4241
 \l__enumext_item_text_viii_box 4436, 4476, 4483
 \l__enumext_item_text_X_box 165
 \l__enumext_item_width_vii_dim . . 3760, 3769,
 3848, 3856, 3857
 \l__enumext_item_width_viii_dim . . 3791, 3800,
 3897, 3905, 3906
 \l__enumext_item_width_X_dim 165
 \l__enumext_itemindent_X_dim 71
 \l__enumext_itemsep_vii_skip 4263
 \l__enumext_itemsep_viii_skip 4508
 \l__enumext_joined_item_aux_vii_int . . 3842,
 3843, 3844, 3845, 3851
 \l__enumext_joined_item_aux_viii_int . 3891,
 3892, 3893, 3894, 3900
 \l__enumext_joined_item_aux_X_int . . . 165
 __enumext_joined_item_vii:w . . 108, 109, 4115,
 4116, 4118, 4118
 \l__enumext_joined_item_vii_int . . 3813, 3814,
 3817, 3819, 3825, 3830, 3835, 3840, 3842, 3848
 __enumext_joined_item_viii:w . 113, 4339, 4340,
 4342, 4342
 \l__enumext_joined_item_viii_int . 3862, 3863,
 3866, 3868, 3874, 3879, 3884, 3889, 3891, 3897
 \l__enumext_joined_item_X_int 165
 \l__enumext_joined_width_vii_dim . 3846, 3853,
 3856, 4224, 4236
 \l__enumext_joined_width_viii_dim 3895, 3902,
 3905, 4458, 4478
 \l__enumext_joined_width_X_dim 165
 __enumext_keyans_addto_prop:n 81, 2742, 2742,
 3120, 3618
 __enumext_keyans_addto_seq:n . 83, 2816, 2816,
 3122, 3620
 __enumext_keyans_addto_seq_link: 2816, 2837,
 2839, 4398
 __enumext_keyans_anspic_code:nnn . 99, 3609,
 3612, 3612
 __enumext_keyans_default_item:n . . 88, 3101,
 3101, 3137
 \l__enumext_keyans_env_bool 34, 3333, 3346, 3502,
 3589
 __enumext_keyans_fake_item: . . 899, 915, 3208
 \l__enumext_keyans_level_h_int . . 28, 642, 669,
 2461, 2523, 2794, 4050, 4292, 4293
 \l__enumext_keyans_level_int . . 28, 1375, 2457,
 2519, 2789, 3501, 3506, 3603
 __enumext_keyans_make_label: 37, 89, 3125, 3141,
 3206
 __enumext_keyans_mini_addvspace: . 52, 1271,
 1271, 3531
 __enumext_keyans_mini_right_cmd:n 54, 1377,
 1406, 1406
 __enumext_keyans_mini_set_vskip: . 51, 1209,
 1209, 1273
 __enumext_keyans_multi_addvspace: 1123, 1134,
 3556
 __enumext_keyans_multi_set_vskip: 49, 1123,
 1123, 1136
 __enumext_keyans_multicols_start: 3519, 3535,
 3537
 __enumext_keyans_multicols_stop: 1410, 3519,
 3562, 3587
 __enumext_keyans_name_and_start: 26, 32, 288,
 288, 3503, 3663, 4297
 __enumext_keyans_parse_keys:n 3462, 3515, 3515
 \l__enumext_keyans_pic_above_int . 139, 3703,
 3704, 3706
 \l__enumext_keyans_pic_above_skip . 100, 139,
 3642, 3682
 __enumext_keyans_pic_arg_two: 100, 3640, 3670,
 3670
 \l__enumext_keyans_pic_below_int . 139, 3703,
 3704, 3707
 \l__enumext_keyans_pic_body_seq . 99–101, 139,
 3607, 3647, 3711
 __enumext_keyans_pic_do:n 101, 3647, 3649, 3695,
 3695, 3699
 \l__enumext_keyans_pic_level_int . . 28, 1359,
 2465, 2527, 2745, 2784, 2819, 2907, 3658, 3659
 __enumext_keyans_pic_row:n . . 101, 3697, 3700,
 3700
 __enumext_keyans_pic_safe_exec: . 100, 3636,
 3656, 3656
 __enumext_keyans_pic_skip_abs:N . 100, 3665,
 3665, 3681
 \l__enumext_keyans_pic_width_dim . 139, 3702,
 3709
 __enumext_keyans_redefine_item: . . 89, 3125,
 3125, 3205
 __enumext_keyans_ref: 41, 694, 712, 3207
 __enumext_keyans_ref:n 40, 691, 694, 694
 __enumext_keyans_safe_exec: . 3461, 3495, 3495
 __enumext_keyans_set_item_width: . 96, 3470,
 3478, 3478
 __enumext_keyans_show_ans: . . 2860, 2868, 2887
 __enumext_keyans_show_item_opt: . 2860, 2875,
 3118, 3632, 4464
 __enumext_keyans_show_left:n . 88, 2860, 2860,
 3116, 3627
 __enumext_keyans_show_pos: . . 2860, 2872, 2900
 __enumext_keyans_starred_item:n . . 88, 3113,
 3113, 3133
 __enumext_keyans_store_ref: . . 82, 2763, 2763,
 3121, 3619, 4396
 __enumext_keyans_store_ref_aux_i: 82, 2763,
 2775, 2778
 __enumext_keyans_store_ref_aux_ii: 82, 2763,
 2804, 2806
 __enumext_keyans_unknown_keys:n . 3039, 3043,
 3047
 __enumext_keyans_unknown_keys:nn 3039, 3049,
 3051
 __enumext_keyans_wrapper_opt:n . . 2011, 2883
 \l__enumext_label_copy_i_tl . . 2269, 2782, 2787,
 2792, 2797
 \l__enumext_label_copy_v_tl 2792
 \l__enumext_label_copy_vi_tl 2787

```

\l__enumext_label_copy_vii_tl 2245, 2256, 2285,
    2782
\l__enumext_label_copy_viii_tl ..... 2797
\l__enumext_label_copy_X_tl ..... 154
\l__enumext_label_fill_left_v_tl ..... 3145
\l__enumext_label_fill_left_X_tl ..... 98
\l__enumext_label_fill_right_v_tl .... 3152
\l__enumext_label_fill_right_X_tl ..... 98
\l__enumext_label_font_style_v_tl 3146, 3631
\l__enumext_label_font_style_vii_tl ... 4212
\l__enumext_label_font_style_viii_tl .. 4446
\l__enumext_label_i_tl ..... 557
\l__enumext_label_ii_tl ..... 557
\l__enumext_label_iii_tl ..... 557
\l__enumext_label_iv_tl ..... 557
\__enumext_label_style:Nnn 26, 36, 471, 471, 486,
    562, 609, 680, 684
\l__enumext_label_v_tl .. 81, 83, 677, 2750, 2824,
    2894, 2934, 3115, 3119, 3465, 3626, 3628
\l__enumext_label_vi_tl . 81, 83, 677, 2747, 2821,
    3626, 3628, 3632
\l__enumext_label_vii_tl . 604, 4138, 4163, 4170
\l__enumext_label_viii_tl 604, 4362, 4390, 4394
\l__enumext_label_width_by_box .. 67, 467, 468
\__enumext_label_width_by_box:Nn 36, 465, 465,
    470, 482, 745
\l__enumext_labelsep_i_dim ... 2892, 2897, 2905,
    2937, 4402, 4417
\l__enumext_labelsep_v_dim ..... 3546
\l__enumext_labelsep_vii_dim . 2369, 2892, 2905,
    3755, 3765, 3849, 4161, 4222, 4238
\l__enumext_labelsep_viii_dim 3786, 3796, 3898,
    4456, 4465, 4480
\l__enumext_labelwidth_i_dim . 2891, 2897, 2904,
    2937, 4402, 4417
\l__enumext_labelwidth_v_dim ..... 3546
\l__enumext_labelwidth_vii_dim ... 2369, 2891,
    2904, 3755, 3764, 3849, 4215, 4219, 4237
\l__enumext_labelwidth_viii_dim .. 3786, 3795,
    3898, 4449, 4453, 4479
\l__enumext_leftmargin_tmp_v_bool . 100, 3672
\l__enumext_leftmargin_tmp_X_bool ..... 71
\l__enumext_leftmargin_tmp_X_dim ..... 71
\l__enumext_leftmargin_X_dim ..... 71
\__enumext_level: 205, 205, 586, 589, 590, 599, 601,
    902, 906, 910, 977, 981, 985, 989, 1072, 1074, 1076,
    1078, 1111, 1113, 1115, 1117, 1121, 1155, 1161, 1166,
    1181, 1384, 1390, 1447, 1449, 1451, 1454, 1461, 1463,
    1465, 1468, 2063, 2065, 2067, 2095, 2096, 2098, 2154,
    2162, 2166, 2170, 2373, 2374, 2949, 2950, 2954, 2955,
    2956, 2964, 2972, 2973, 2976, 2983, 2984, 2988, 2991,
    2993, 3011, 3012, 3013, 3016, 3019, 3269, 3271, 3290,
    3295, 3339, 3352, 3359, 3370, 3372, 3375, 3376, 3378,
    3382, 3389, 3392, 3394, 3396, 3397, 3398, 3399, 3402,
    3407, 3413, 3419, 3422, 3424, 3430, 3442, 3444
\l__enumext_level_h_int 107, 28, 238, 261, 275, 625,
    662, 1366, 1879, 1899, 2264, 2497, 2509, 3347, 4045,
    4046
\l__enumext_level_int . 92, 28, 207, 248, 260, 276,
    370, 1084, 1187, 1365, 1873, 1905, 2241, 2251, 2257,
    2263, 2270, 2279, 2284, 2496, 2508, 2724, 3221, 3303,
    3304, 3315, 3323, 3337, 3350, 3403, 3510, 3599, 4089,
    4099, 4305, 5150, 5154, 5160, 5164
\__enumext_list_arg_two_ii: ..... 3187
\__enumext_list_arg_two_iii: ..... 3187
\__enumext_list_arg_two_iv: ..... 3187
\__enumext_list_arg_two_v: . 89, 3187, 3467, 3673
\__enumext_list_arg_two_vii: ..... 3227, 4026
\__enumext_list_arg_two_viii: .... 3227, 4275
\l__enumext_listoffset_v_dim . 3486, 3491, 3548
\l__enumext_listparindent_vii_dim .... 4225
\l__enumext_listparindent_viii_dim ... 4459
\__enumext_log_answer_vars: . 33, 342, 350, 2731
\__enumext_log_global_vars: . 33, 342, 342, 2730
\__enumext_make_label ..... 2996
\__enumext_make_label: ..... 37, 86, 3007, 3216
\l__enumext_mark_answer_sym_tl 71, 2017, 2220,
    2386, 2909, 2922, 4406
\l__enumext_mark_position_str 128, 2021, 2022,
    2048, 2049, 2218
\l__enumext_mark_ref_sym_tl .. 2034, 2359, 2851
\l__enumext_meta_path_tl . 124, 4738, 4739, 4741,
    4742
\c__enumext_meta_paths_prop ..... 121, 4714
\__enumext_mini_addvspace_vii: 53, 1345, 1345,
    3923
\__enumext_mini_addvspace_viii: 53, 1345, 1351,
    3978
__enumext_mini_env* ..... 368
\__enumext_mini_right_cmd:n 54, 1379, 1381, 1381
\__enumext_mini_set_vskip_vii: 52, 1288, 1288,
    1347
\__enumext_mini_set_vskip_viii: 52, 1288, 1310,
    1353
\__enumext_minipage:w 33, 363, 365, 374, 3709, 4224,
    4458
\l__enumext_minipage_active_v_bool 3529, 3554,
    3567, 3575
\g__enumext_minipage_active_vii_bool .. 105,
    3934, 3939, 3958
\l__enumext_minipage_active_vii_bool . 3919,
    3930
\g__enumext_minipage_active_viii_bool 3989,
    3994, 4013
\l__enumext_minipage_active_viii_bool 3974,
    3985
\g__enumext_minipage_active_X_bool ... 165
\l__enumext_minipage_active_X_bool .... 86
\__enumext_minipage_add_space: .. 50, 94, 1151,
    1172, 3380
\g__enumext_minipage_after_skip 86, 1292, 1304,
    3956, 4011
\l__enumext_minipage_after_skip .. 50, 95, 86,
    1164, 1192, 1198, 1204, 1211, 1220, 1223, 1235, 1253,
    1264, 1280, 1312, 1325, 1339, 3445, 3584
\g__enumext_minipage_center_vii_bool . 3943,
    3959
\g__enumext_minipage_center_viii_bool 3998,
    4014
\g__enumext_minipage_center_X_bool ... 165
\l__enumext_minipage_hsep_v_dim ..... 3527
\l__enumext_minipage_hsep_vii_dim .... 3917
\l__enumext_minipage_hsep_viii_dim ... 3972
\l__enumext_minipage_left_skip 86, 1212, 1218,
    1227, 1244, 1256, 1276, 1286, 1290, 1295, 1299, 1313,
    1317, 1331, 1349, 1355
\l__enumext_minipage_left_v_dim .. 3525, 3533

```

```

\l__enumext_minipage_left_vii_dim 3913, 3925
\l__enumext_minipage_left_viii_dim 3968, 3980
\l__enumext_minipage_left_X_dim ..... 86
\g__enumext_minipage_right_skip 86, 1291, 1296,
    1300, 3942, 3997
\l__enumext_minipage_right_skip . 50, 86, 1153,
    1159, 1164, 1169, 1213, 1219, 1231, 1249, 1260, 1314,
    1321, 1335, 1392, 1414
\l__enumext_minipage_right_v_dim . 1408, 1413,
    3523, 3527
\g__enumext_minipage_right_vii_dim 104, 3921,
    3941, 3961
\l__enumext_minipage_right_vii_dim 104, 3911,
    3916, 3922
\g__enumext_minipage_right_viii_dim .. 3976,
    3996, 4016
\l__enumext_minipage_right_viii_dim .. 3966,
    3971, 3977
\g__enumext_minipage_right_X_dim ..... 165
\g__enumext_minipage_right_X_skip .... 165
\__enumext_minipage_set_skip: . 50, 1151, 1151,
    1174
\g__enumext_minipage_stat_int 94, 86, 1397, 1419,
    3379, 3432, 3437, 3530, 3577, 3582
\l__enumext_miniright_code_vii_box 3950, 3954
\g__enumext_miniright_code_vii_tl 105, 3945,
    3952, 3960
\l__enumext_miniright_code_viii_box .. 4005,
    4009
\g__enumext_miniright_code_viii_tl 4000, 4007,
    4015
\l__enumext_miniright_code_X_box ..... 165
\__enumext_multi_addvspace: . 49, 95, 1106, 1106,
    3410
\__enumext_multi_set_vskip: 48, 1070, 1070, 1108
\l__enumext_multicols_above_ii_skip ... 1089
\l__enumext_multicols_above_iii_skip .. 1095
\l__enumext_multicols_above_iv_skip ... 1101
\l__enumext_multicols_above_v_skip 1125, 1139,
    1149
\l__enumext_multicols_above_X_skip .... 79
\l__enumext_multicols_below_v_skip 1129, 1143,
    3569
\l__enumext_multicols_below_X_skip .... 79
\g__enumext_multicols_right_X_skip .... 79
\__enumext_multicols_start: 94, 3384, 3386, 3386
\__enumext_multicols_stop: 95, 1386, 3416, 3416,
    3450
\__enumext_nested_base_line_fix: . 43, 93, 107,
    819, 829, 3319, 4066
\__enumext_newlabel:nn 29, 35, 72, 422, 422, 2295,
    2810
\l__enumext_newlabel_arg_one_tl 29, 35, 72, 82,
    154, 2288, 2296, 2358, 2799, 2811, 2849
\l__enumext_newlabel_arg_two_tl 29, 35, 71, 154,
    2244, 2254, 2267, 2282, 2297, 2786, 2791, 2796, 2812
\__enumext_parse_foreach_keys:n .. 4763, 4779,
    4796
\__enumext_parse_foreach_keys:nn . 4763, 4786,
    4798
\__enumext_parse_keys:n 43, 58, 3265, 3310, 3310
\__enumext_parse_keys_vii:n . 43, 58, 4021, 4058,
    4058
\__enumext_parse_keys_viii:n . 4271, 4310, 4310

\__enumext_parse_save_key:n 68, 2088, 2093, 2093
\__enumext_parse_save_key_vii:n 68, 2083, 2093,
    2101
\__enumext_parse_series:n 58, 93, 107, 1576, 1576,
    3318, 4064
\__enumext_parse_store_keys:n ..... 93
\l__enumext_parsep_i_skip 1087, 1089, 1190, 1221
\l__enumext_parsep_ii_skip ... 1093, 1095, 1196
\l__enumext_parsep_iii_skip .. 1099, 1101, 1202
\l__enumext_parsep_vii_skip ..... 4226
\l__enumext_parsep_viii_skip ..... 4460
\l__enumext_partopsep_v_skip . 1141, 1145, 1247,
    1251, 1258, 1262, 1278, 1282
\l__enumext_partopsep_viii_skip ..... 1323
\__enumext_phantomsection: 35, 386, 415, 419, 435
\__enumext_print_footnote: ... 3716, 3739, 4244,
    4486
\__enumext_print_keyans_box:NN 71, 2212, 2212,
    2225, 2369, 2372, 2896, 2936, 4402, 4417
\l__enumext_print_keyans_i_tl .... 4548, 4570
\l__enumext_print_keyans_ii_tl ... 4552, 4571
\l__enumext_print_keyans_iii_tl .. 4556, 4572
\l__enumext_print_keyans_iv_tl ... 4560, 4573
\l__enumext_print_keyans_starred_tl 117, 118,
    128, 4544, 4592
\l__enumext_print_keyans_vii_tl 117, 4564, 4574
\l__enumext_print_keyans_X_tl ..... 128
\__enumext_printkeyans:nnn 118, 4575, 4578, 4578
\__enumext_redefine_item: . 86, 2996, 2996, 3215
\l__enumext_ref_key_arg_tl 38, 50, 220, 579, 580,
    593, 624, 627, 638, 644, 655, 696, 697, 708
\l__enumext_ref_the_count_tl . 38, 50, 586, 589,
    592, 632, 634, 637, 649, 651, 654, 702, 704, 707
\__enumext_regex_counter_style: .. 30, 38, 215,
    215, 587, 633, 650, 703
\__enumext_register_counter_style:Nn .. 455,
    455, 460, 461, 462, 463, 464
\__enumext_remove_extra_parsep_vii: .. 4036,
    4253, 4253
\__enumext_remove_extra_parsep_viii: . 4285,
    4496, 4496
\__enumext_renew_footnote: ... 3716, 3720, 4196,
    4439
\l__enumext_renew_the_count_v_tl 705, 714, 716
\l__enumext_renew_the_count_vii_tl 635, 664,
    666
\l__enumext_renew_the_count_viii_tl 652, 671,
    673
\l__enumext_renew_the_count_X_tl ..... 50
\__enumext_rescan_anskey_env:n .. 78, 80, 2606,
    2701, 2709
\__enumext_reset_global_bool: .. 318, 321, 330
\__enumext_reset_global_int: ... 318, 320, 324
\__enumext_reset_global_tl: .... 318, 322, 336
\__enumext_reset_global_vars: . 32, 81, 318, 318,
    2739
\l__enumext_resume_active_bool 58, 60, 61, 1580,
    1700
\__enumext_resume_counter: .. 60, 61, 1698, 1704,
    1711
\__enumext_resume_counter:n . 58, 60, 1669, 1674,
    1698, 1698, 1768, 1776
\__enumext_resume_counter_save_ans: .. 60, 61,
    1698, 1709, 1741

```

```

\__enumext_resume_counter_series: 60, 61, 1698,
    1707, 1724
\g__enumext_resume_int ... 61, 1621, 1715, 1716
\__enumext_resume_last:n .. 58, 1576, 1582, 1595
\l__enumext_resume_name_tl 61, 1617, 1625, 1628,
    1644, 1652, 1655, 1701, 1702, 1730, 1737
\__enumext_resume_save_counter: .. 59, 96, 108,
    1608, 1608, 3456, 4083
\__enumext_resume_series:n . 60, 1544, 1665, 1665
\__enumext_resume_starred: . 61, 1545, 1762, 1762
\g__enumext_resume_vii_int 61, 1648, 1720, 1721
\l__enumext_rightmargin_vii_dim .. 3767, 3771,
    3776
\l__enumext_rightmargin_viii_dim . 3798, 3802,
    3807
\__enumext_safe_exec: .. 34, 92, 3264, 3299, 3299
\__enumext_safe_exec_vii: . 34, 4020, 4041, 4041
\__enumext_safe_exec_viii: ... 4270, 4290, 4290
\l__enumext_series_name_tl ..... 60
\l__enumext_series_str .. 59, 93, 107, 1542, 1578,
    1586, 1587, 1589, 1591, 1612, 1615, 1619, 1639, 1642,
    1646, 3314, 4062
\__enumext_set_error:nn ..... 4700, 4710, 4712
\__enumext_set_item_width: . 92, 3274, 3282, 3282
\__enumext_set_parse:n ..... 4684, 4700, 4700
\l__enumext_setkey_tmpa_int ... 119, 4677, 4681
\l__enumext_setkey_tmpa_seq .. 119, 4675, 4685,
    4691, 4693, 4695, 4707
\l__enumext_setkey_tmpa_tl ... 119, 4683, 4687
\l__enumext_setkey_tmpb_seq .. 119, 4676, 4679,
    4683, 4684
\l__enumext_setkey_tmpb_tl 119, 4702, 4704, 4705
\l__enumext_show_answer_bool . 2028, 2052, 2380,
    2866, 2880, 3623, 4400
\__enumext_show_length:nnn .. 45, 223, 223, 4921,
    4922, 4923, 4924, 4925, 4926, 4927, 4928, 4929, 4930,
    4936, 4937, 4938, 4939, 4940, 4941, 4942, 4943, 4944,
    4945
\l__enumext_show_position_bool ... 2031, 2055,
    2384, 2870, 2881, 3624, 4404
\g__enumext_standar_bool 31, 92, 34, 237, 240, 259,
    333, 1610, 1675, 1687, 1713, 1726, 1764, 1904, 1918,
    2249, 2262, 2277, 3334
\l__enumext_standar_bool . 92, 96, 34, 2250, 3306,
    3455, 4055
\l__enumext_standar_first_bool 31, 93, 34, 264,
    832, 1597, 1744, 1806, 1813
\__enumext_standar_item_vii:w . 109, 4123, 4125,
    4125
\__enumext_standar_item_viii:w 113, 114, 4347,
    4349, 4349
\__enumext_standar_ref: .... 39, 577, 597, 3217
\__enumext_standar_ref:n .... 38, 569, 577, 577
\g__enumext_standar_series_tl . 61, 1599, 1600,
    1766, 1769
\__enumext_standar_unknown_keys:n 3079, 3083,
    3087
\__enumext_standar_unknown_keys:nn 3079, 3089,
    3091
\g__enumext_starred_bool 31, 107, 34, 247, 250, 274,
    334, 1637, 1680, 1691, 1718, 1733, 1772, 1878, 1924,
    2240, 2780, 3962
\l__enumext_starred_bool 107, 108, 34, 1371, 2278,
    2313, 2319, 2367, 2655, 2660, 2889, 2902, 3307, 4054,
    4082, 4298, 4302
\__enumext_starred_columns_set_vii: .. 3749,
    3749, 4029
\__enumext_starred_columns_set_viii: . 3749,
    3780, 4278
\l__enumext_starred_first_bool 31, 107, 34, 279,
    843, 1602, 1753, 1806, 1813
\__enumext_starred_item:nn ... 2959, 2959, 3002
\__enumext_starred_item_exec: 114, 4392, 4392,
    4443
\__enumext_starred_item_vii:w . 109, 4122, 4141,
    4141
\__enumext_starred_item_vii_aux_i:w .. 4141,
    4146, 4149
\__enumext_starred_item_vii_aux_ii:w . 4141,
    4147, 4152, 4154
\__enumext_starred_item_vii_aux_iii:w 4141,
    4157, 4166
\__enumext_starred_item_viii:w 113, 114, 4346,
    4365, 4365
\__enumext_starred_item_viii_aux_i:w .. 114,
    4365, 4370, 4373
\__enumext_starred_item_viii_aux_ii:w . 114,
    4365, 4371, 4385, 4387
\__enumext_starred_joined_item_vii:n 103, 109,
    3811, 3811, 4120
\__enumext_starred_joined_item_viii:n . 103,
    113, 3811, 3860, 4344
\__enumext_starred_ref: .... 40, 622, 660, 3247
\__enumext_starred_ref:n .... 39, 616, 622, 622
\g__enumext_starred_series_tl . 61, 1604, 1605,
    1774, 1777
\__enumext_starred_unknown_keys:n 3061, 3063,
    3065
\__enumext_starred_unknown_keys:nn 3061, 3067,
    3069
\__enumext_start_from:NNn 41, 719, 719, 732, 754,
    760
\l__enumext_start_i_int ..... 1716, 1728, 1747
\__enumext_start_item_tmp_vii: 106, 4032, 4105,
    4105
\__enumext_start_item_tmp_viii: .. 112, 4281,
    4329, 4329
\__enumext_start_item_vii:w 109, 110, 4133, 4138,
    4163, 4170, 4172, 4172
\__enumext_start_item_viii:w . 114, 4357, 4362,
    4390, 4420, 4420
\g__enumext_start_line_tl 31, 34, 267, 282, 339,
    1948, 1953, 1958, 1972, 1977, 1982
\__enumext_start_list:nn .. 33, 89, 100, 357, 359,
    3268, 3464, 3637, 4024, 4273
\__enumext_start_mini_vii: 107, 3909, 3909, 4074
\__enumext_start_mini_viii: ... 112, 3964, 3964,
    4321
\__enumext_start_save_ans_msg: 62, 1790, 1790,
    1815
\__enumext_start_store_level: . 93, 3267, 3328,
    3328
\__enumext_start_store_level_vii: 108, 4023,
    4085, 4085
\l__enumext_start_vii_int ... 1721, 1735, 1756
\l__enumext_start_X_int ..... 98
\__enumext_stop_item_tmp_vii: .. 106, 108, 110,
    4031, 4035, 4107, 4174

```

```

\__enumext_stop_item_tmp_viii: 112, 113, 4280,
    4284, 4331, 4422
\__enumext_stop_item_vii: 110, 111, 4174, 4229,
    4229
\__enumext_stop_item_viii: 116, 4422, 4471, 4471
\__enumext_stop_list: .. 33, 357, 360, 3278, 3475,
    3650, 4037, 4287
\__enumext_stop_mini_vii: 105, 108, 3909, 3928,
    4078
\__enumext_stop_mini_viii: 113, 3964, 3983, 4325
\__enumext_stop_save_ans_msg: . 62, 1790, 1795,
    2728
\__enumext_stop_store_level: .. 93, 3279, 3328,
    3357
\__enumext_stop_store_level_vii: . 108, 4038,
    4085, 4095
\l__enumext_store_active_bool 28, 63, 110, 1745,
    1754, 1822, 2453, 3332, 3345, 3497, 3505, 3595, 3654,
    4087, 4097, 4304
\__enumext_store_active_keys:n .. 68, 93, 2061,
    2061, 3325
\__enumext_store_active_keys_vii:n . 68, 107,
    2061, 2071, 4065
\__enumext_store_addto_prop:n 69, 81, 2136, 2136,
    2144, 2304, 2761, 4395
\__enumext_store_addto_seq:n 70, 83, 2145, 2145,
    2149, 2156, 2170, 2178, 2187, 2201, 2209, 2362, 2854
\l__enumext_store_anskey_arg_tl .. 28, 73, 110,
    2310, 2315, 2317, 2322, 2329, 2332, 2342, 2347, 2350,
    2356, 2362
\__enumext_store_anskey_code:n 73, 75, 80, 2301,
    2301, 2446, 2699, 2707
\l__enumext_store_anskey_env_tl .. 28, 79, 110,
    2629, 2633, 2639, 2701, 2709
\l__enumext_store_anskey_opt_tl 28, 79, 80, 110,
    2630, 2657, 2663, 2670, 2676, 2686, 2696, 2705
\__enumext_store_anskey_safe_outer: .... 76
\g__enumext_store_columns_break_bool . 2553,
    2654, 2716
\l__enumext_store_columns_break_bool . 2312,
    2402
\l__enumext_store_current_label_tl 28, 81, 83,
    114, 110, 2744, 2747, 2750, 2757, 2759, 2761, 2818,
    2821, 2824, 2830, 2835, 2845, 2854, 4375, 4380, 4381,
    4394, 4395, 4397
\l__enumext_store_current_label_tmp_tl . 28,
    110, 3115, 3119
\l__enumext_store_current_opt_arg_tl 28, 114,
    110, 2864, 2877, 2883, 4383
\__enumext_store_internal_ref: .. 71, 73, 2226,
    2226, 2307
\g__enumext_store_item_join_int .. 2556, 2661,
    2665, 2717
\l__enumext_store_item_join_int .. 2320, 2324,
    2405
\g__enumext_store_item_star_bool . 2558, 2668,
    2718
\l__enumext_store_item_star_bool . 2327, 2407
\g__enumext_store_item_symbol_sep_dim 2563,
    2683, 2688, 2720
\l__enumext_store_item_symbol_sep_dim 2339,
    2344, 2412
\g__enumext_store_item_symbol_tl . 2561, 2674,
    2678, 2719
\l__enumext_store_item_symbol_tl . 2330, 2334,
    2410
\l__enumext_store_keyans_item_opt_sep_
    tl .... 2014, 2755, 2757, 2828, 2832, 4378, 4380
\__enumext_store_level_close: . 70, 2150, 2174,
    3361
\__enumext_store_level_close_vii: . 70, 2181,
    2205, 4101
\__enumext_store_level_open: 70, 93, 2150, 2150,
    3340, 3353
\__enumext_store_level_open_vii: .. 70, 2181,
    2181, 4091
\g__enumext_store_name_tl 28, 63, 110, 338, 345,
    346, 347, 348, 1798, 1824, 1947, 1952, 1957, 1971,
    1976, 1981, 2726
\l__enumext_store_name_tl 28, 62, 64, 110, 1631,
    1634, 1658, 1661, 1749, 1758, 1793, 1802, 1803, 1824,
    1825, 1826, 1828, 1829, 1831, 1833, 1834, 1836, 1838,
    1839, 1863, 2138, 2140, 2147, 2290, 2291, 2392, 2635,
    2801, 2802, 2915, 2928, 4412
\l__enumext_store_ref_key_bool 73, 2037, 2305,
    2353, 2765, 2842
\l__enumext_store_save_key_vii_bool .. 2073,
    2103
\l__enumext_store_save_key_vii_tl 2075, 2076,
    2104, 2105, 2185, 2193, 2197, 2201
\l__enumext_store_save_key_X_bool .. 68, 128
\l__enumext_store_save_key_X_tl .... 68, 128
\l__enumext_store_upper_level_X_bool .. 128
\__enumext_storing_exec: . 62, 63, 77, 1800, 1816,
    1820
\__enumext_storing_set:n .. 62, 1785, 1800, 1800
\l__enumext_the_counter_v_tl ..... 704
\l__enumext_the_counter_vii_tl ..... 634
\l__enumext_the_counter_viii_tl ..... 651
\l__enumext_the_counter_X_tl ..... 50
\__enumext_tmp:n 45, 49, 54, 60, 71, 78, 79, 85, 92, 97,
    98, 109, 131, 138, 157, 161, 165, 185, 819, 828, 1538,
    1549, 1781, 1789, 1842, 1860, 2001, 2042, 2043, 2060,
    2079, 2092, 2228, 2235, 2236, 2257, 2270, 2273, 2284,
    2767, 2774, 3039, 3046, 3079, 3086, 3187, 3226, 3227,
    3261
\__enumext_tmp:nn 487, 508, 509, 540, 541, 556, 749,
    774, 855, 877, 878, 898, 951, 959, 960, 974, 1039, 1055,
    1056, 1069, 1427, 1443, 3023, 3038
\__enumext_tmp:nnn 557, 573, 574, 575, 576, 604, 620,
    621
\__enumext_tmp:nnnnnn 775, 800, 803, 806, 808, 810,
    813, 816
\__enumext_tmp:w ..... 4523, 4526
\l__enumext_tmpa_vii_int 3759, 3762, 3771, 3802
\l__enumext_tmpa_viii_int ..... 3790, 3793
\l__enumext_tmpa_X_dim ..... 165
\l__enumext_tmpa_X_int ..... 165
\l__enumext_topsep_v_skip 1127, 1131, 1216, 1229,
    1237, 1242, 1262, 1266, 3653, 3685
\l__enumext_topsep_vii_skip .. 1293, 1302, 1306
\l__enumext_topsep_viii_skip . 1315, 1337, 1341
\__enumext_undefine_anskey_env: . 76, 81, 2486,
    2486, 2737
\l__enumext_vspace_a_star_v_bool ..... 1476
\l__enumext_vspace_a_star_vii_bool ... 1498
\l__enumext_vspace_a_star_viii_bool ... 1509
\l__enumext_vspace_a_star_X_bool ..... 98
\__enumext_vspace_above: 55, 94, 1444, 1444, 3366
\__enumext_vspace_above_v: . 56, 1472, 1472, 3521

```


<code>\l__enumext_vspace_above_v_skip</code> .. 1474, 1478, 1480	<code>67–77, 79, 81, 82, 85, 86, 88–90, 92, 93, 96, 97, 100, 102, 104, 107, 108, 117–119, 121, 124, 125, 127</code>
<code>__enumext_vspace_above_vii:</code> 56, 107, 1494, 1494, 4071	<code>keyans*</code> 25, 26, 28–32, 36, 39–45, 47, 52, 53, 56, 57, 63, 64, 66, 67, 69, 77, 82, 87, 91, 101, 102, 104, 112, 124, 126, 129
<code>\l__enumext_vspace_above_vii_skip</code> 1496, 1500, 1502	<code>keyanspic</code> 25, 26, 28, 29, 32, 36, 37, 40, 63, 64, 66, 69, 70, 77, 81–83, 98–100, 126
<code>__enumext_vspace_above_viii:</code> . 56, 1494, 1505, 4319	<code>keyans</code> 25, 26, 28, 29, 31, 32, 36, 37, 40–43, 45, 47, 49, 51, 52, 54–56, 63, 64, 66, 67, 69, 70, 77, 81–83, 87–90, 96–100, 104, 113, 124, 126
<code>\l__enumext_vspace_above_viii_skip</code> 1507, 1511, 1513	Environments:
<code>\l__enumext_vspace_b_star_v_bool</code> 1487	<code>list</code> 30, 33, 89, 92
<code>\l__enumext_vspace_b_star_vii_bool</code> ... 1520	<code>lrbox</code> 102, 110, 111, 115, 116
<code>\l__enumext_vspace_b_star_viii_bool</code> ... 1531	<code>minipage</code> . 30, 33, 34, 47, 49, 50, 98–102, 110, 111, 116
<code>\l__enumext_vspace_b_star_X_bool</code> 98	<code>multicols</code> 47–50, 54, 94, 95
<code>__enumext_vspace_below:</code> 56, 96, 1458, 1458, 3454	<code>scontents</code> 77, 79
<code>__enumext_vspace_below_v:</code> . 56, 1483, 1483, 3591	exp commands:
<code>\l__enumext_vspace_below_v_skip</code> .. 1485, 1489, 1491	<code>\exp_after:wN</code> 4526
<code>__enumext_vspace_below_vii:</code> 57, 108, 1516, 1516, 4081	<code>\exp_args:Ne</code> 2698, 2706, 3322, 4514
<code>\l__enumext_vspace_below_vii_skip</code> 1518, 1522, 1524	<code>\exp_args:NV</code> ... 2418, 2573, 3049, 3067, 3089, 4798
<code>__enumext_vspace_below_viii:</code> . 57, 1516, 1527, 4327	<code>\exp_not:N</code> . 58, 478, 592, 637, 654, 707, 908, 922, 923, 934, 935, 946, 947, 2358, 2389, 2390, 2847, 2912, 2913, 2925, 2926, 4409, 4410, 4523
<code>\l__enumext_vspace_below_viii_skip</code> 1529, 1533, 1535	<code>\exp_not:n</code> 269, 284, 297, 305, 313, 531, 551, 592, 593, 637, 638, 654, 655, 707, 708, 909, 1565, 1574, 2025, 2122, 2134, 2296, 2324, 2334, 2344, 2358, 2359, 2665, 2678, 2688, 2811, 2849, 2851, 4627, 4637, 4830, 4835
<code>__enumext_widest_from:nNNn</code> .. 41, 733, 733, 748, 767	F
<code>\g__enumext_widest_label_tl</code> 26, 36, 67, 475, 479, 483	<code>\fbox</code> 2008
<code>\l__enumext_wrap_label_opt_v_bool</code> 3109	<code>\fboxrule</code> 2008
<code>\l__enumext_wrap_label_opt_vii_bool</code> 109, 4132	<code>\fboxsep</code> 2008
<code>\l__enumext_wrap_label_opt_viii_bool</code> .. 114, 4356	file commands:
<code>\l__enumext_wrap_label_opt_X_bool</code> 98	<code>\file_input_stop:</code> 5234
<code>\l__enumext_wrap_label_v_bool</code> 3105, 3109, 3117, 3147	<code>first</code> 960
<code>\l__enumext_wrap_label_vii_bool</code> .. 109, 4131, 4136, 4144, 4213	<code>font</code> 487
<code>\l__enumext_wrap_label_viii_bool</code> . 114, 4355, 4360, 4368, 4447	<code>\footnote</code> 101
<code>\l__enumext_wrap_label_X_bool</code> 98	<code>\footnote</code> 101, 3724
<code>__enumext_wrapper_label_v:n</code> 3149, 3632	<code>\footnotemark</code> 3734
<code>__enumext_wrapper_label_vii:n</code> 4216	<code>\footnotesize</code> 2390, 2913, 2926, 4410
<code>__enumext_wrapper_label_viii:n</code> 4450	<code>\footnotetext</code> 3718
<code>\l__enumext_write_aux_file_tl</code> . 29, 72, 82, 154, 2293, 2299, 2808, 2814	<code>\foreachkeyans</code> 16, 121, 4763
<code>__enumext_zero_parsep:</code> 50, 1185, 1185	G
<code>enumext*</code> 5, 4018	<code>\getkeyans</code> 16, 117, 4512
<code>enumXi</code> 447	group commands:
<code>enumXii</code> 447	<code>\group_begin:</code> .. 2388, 2433, 2608, 2695, 2911, 2924, 4192, 4211, 4408, 4435, 4445, 4569
<code>enumXiii</code> 447	<code>\group_end:</code> 2395, 2449, 2712, 2918, 2931, 4221, 4233, 4415, 4455, 4475, 4576
<code>enumXiv</code> 447	H
<code>enumXv</code> 447	<code>\hbadness</code> 4240, 4482
<code>enumXvi</code> 447	hbox commands:
<code>enumXvii</code> 447	<code>\hbox_set:Nn</code> 467
<code>enumXviii</code> 447	<code>\hfill</code> 517, 521, 526, 527, 1388, 1412, 2358, 2847, 3933, 3988
Environments provide by enumext :	hook commands:
<code>anskey*</code> 28, 63, 71, 72, 74, 76, 77, 79, 81, 93, 108, 118, 123, 125	<code>\hook_gput_code:nnn</code> 9, 195, 199, 203, 384
<code>enumext*</code> 25, 26, 29–31, 34, 36, 39–45, 47, 52, 53, 56–62, 64, 65, 67–77, 79–82, 86, 87, 91–93, 101–103, 105, 108, 110, 111, 113, 115, 117–119, 121, 124, 127, 129	<code>\hook_gremove_code:nn</code> 79, 2624
<code>enumext</code> . 25, 26, 30, 31, 34, 36–43, 45–52, 54–62, 64, 65,	<code>\hook_gset_rule:nnnn</code> 385
	<code>\hook_if_empty:nTF</code> 2622
	<code>\hspace</code> 4251, 4494
	<code>\hyperlink</code> 73, 83
	<code>\hyperlink</code> 2358, 2847
	<code>\hypertarget</code> 35
	<code>\hypertarget</code> 414

I

\IfHyperBoolean 392

\IfPackageLoadedTF 11, 19, 388, 402

\ignorespaces 911

\inputlineno 269, 284, 297, 305, 313

int commands:

 \int_add:Nn 3844, 3893

 \int_case:nn ... 1084, 1187, 1873, 1899, 1938, 1962

 \int_compare:nNnTF .. 370, 625, 642, 662, 669, 1165, 1214, 1359, 1375, 1986, 1992, 2457, 2461, 2465, 2473, 2519, 2523, 2527, 2724, 2745, 2784, 2789, 2794, 2819, 2907, 3304, 3315, 3337, 3350, 3388, 3403, 3418, 3432, 3441, 3506, 3510, 3539, 3564, 3577, 3599, 3603, 3659, 3814, 3824, 3840, 3863, 3873, 3889, 4046, 4050, 4089, 4099, 4246, 4255, 4293, 4305, 4488, 4498, 4681, 4813

 \int_compare_p:nNn ... 238, 248, 260, 261, 275, 276, 1365, 1366, 1879, 1905, 2241, 2251, 2263, 2264, 2279, 2320, 2496, 2497, 2508, 2509, 2661, 3347

 \int_decr:N 3843, 3892

 \int_eval:n .. 355, 762, 2140, 2291, 2390, 2802, 2913, 2926, 3202, 3246, 3832, 3881, 4410

 \int_from_alph:n 727, 741

 \int_from_roman:n 729, 743

 \int_gadd:Nn 3845, 3894

 \int_gdecr:N 1882, 1887, 1891, 1895, 1908

 \int_gincr:N 1715, 1720, 2303, 2857, 2946, 2980, 3123, 3379, 3530, 3621, 4109, 4187, 4333, 4399

 \int_gset:Nn 1931, 3732

 \int_gset_eq:NN 1614, 1621, 1627, 1633, 1641, 1648, 1654, 1660, 3729

 \int_gzero:N . 326, 327, 328, 1397, 1419, 1998, 2717, 3437, 3582, 4264, 4509

 \int_if_exist:NnTF 1589, 1625, 1631, 1652, 1658, 1836

 \int_incr:N 2472, 3303, 3501, 3658, 4045, 4108, 4292, 4332

 \int_mod:nn 4257, 4500

 \int_new:N . 28, 29, 30, 31, 32, 33, 61, 62, 86, 102, 121, 141, 142, 147, 148, 149, 151, 162, 168, 169, 170, 171, 172, 1591, 1839

 \int_set:Nn 723, 727, 729, 1728, 1735, 1747, 1756, 2609, 3703, 3704, 3759, 3790, 3813, 3819, 3835, 3862, 3868, 3884, 4240, 4482, 4677, 4815

 \int_set_eq:NN 1716, 1721, 3842, 3891

 \int_sign:n 1933

 \int_step_function:nnN 2257, 2270, 2284

 \int_step_function:nnnN 4819

 \int_step_inline:nn 4729

 \int_step_inline:nnn 3705

 \int_to_roman:n 207, 2237, 2274

 \int_use:N 348, 353, 354, 1166, 1730, 1737, 1749, 1758, 3202, 3221, 3246, 3323, 3389, 3398, 3413, 3419, 3442, 3817, 3818, 3830, 3866, 3867, 3879, 5150, 5154, 5160, 5164

 \int_zero:N 4249, 4492

\item . 85, 88, 108, 110, 113, 115, 361, 2158, 2164, 2189, 2195, 2317, 2821, 2824, 2998, 3127, 3690, 4030, 4032, 4279, 4281, 4397

\item* 5, 14, 66, 3125

item-pos* 3023

item-sym* 3023

\itemindent 90

\itemindent 89

itemindent 855

\itemsep 99, 100

\itemsep 3674, 3680

\itemwidth . 437, 2008, 3284, 3293, 3480, 3489, 3853, 3857, 3902, 3906

K

keyans 14, 3459

keyans* 14, 4268

keyanspic 15, 3634

Keys for \anskey provide by enumext:

 break-col 73, 75, 78, 79

 item-join 73, 75, 78, 80

 item-pos* 73, 75, 78, 80

 item-star 73, 75, 78, 80

 item-sym* 73, 75, 78, 80

Keys for anskey* provide by enumext:

 break-col 73, 75, 78, 79

 item-join 73, 75, 78, 80

 item-pos* 73, 75, 78, 80

 item-star 73, 75, 78, 80

 item-sym* 73, 75, 78, 80

Keys for environments provide by enumext:

 above* 27, 55, 56, 94, 107

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

 after 45, 46, 96, 108, 113

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

 base-fix 43, 58, 69, 93, 107, 118

 before* 45, 46, 94, 107, 112

 before 45, 46

 below* 27, 55-57, 96, 108

 below 27, 55-57, 96, 108, 113

 check-ans .. 29-31, 62-66, 69, 81, 83, 96, 108, 111, 125

 columns-sep 47, 94

 columns 27, 47, 55, 94

 first 45, 46, 110

 font 37, 86, 89, 110

 item-pos* 85, 86

 item-sym* 28, 85, 86

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

 itemsep 42, 91

 labelsep 37, 90, 110

 labelwidth 36-41, 90

 label 26, 36, 38, 41, 102

 lisparindent 91

 list-indent 27, 43, 44, 100

 list-offset 43, 44, 92, 96

 listparindent 43, 110

 mark-ans 67, 69, 74

 mark-pos 67, 123

 mark-ref 67, 69, 71, 73

 mini-env 27, 34, 47, 54, 55, 69, 94, 104, 105, 107, 108, 112

 mini-right* 27, 30, 47, 69, 105, 107, 108

 mini-right 27, 30, 47, 53, 69, 105, 107, 108

 mini-sep 27, 47, 69, 94

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

 noitemsep 42, 50

 nosep 42, 50

 parindent 91

 parsep 42, 91, 110

 partopsep 42

 ref 26, 30, 38-40, 124

 resume* 26, 57, 58, 61-63, 69, 96, 108, 119

 resume 26, 33, 57-63, 69, 96, 108, 119

 rightmargin 43, 102

 save-ans 28, 33, 58-62, 64, 65, 68-70, 75-77, 81, 83, 88, 97-99, 113, 114, 117, 119, 125

 save-key 28, 58, 68, 93, 107

save-pos	69
save-ref	29, 35, 67, 69, 71, 73, 82, 83, 88, 114
save-sep	67, 69, 114
series	26, 57–61, 69, 93, 96, 107, 108, 119
show-ans	67, 69, 71, 73, 74, 88, 114
show-length	31, 45, 124
show-pos	28, 67, 71, 73, 74, 83, 88, 114
start*	27, 41, 58
start	27, 30, 41, 58
store-key	68
topsep	42
widest	26, 30, 41
wrap-ans	35, 67, 69, 71, 74
wrap-label*	27, 37, 85, 86, 89, 109, 110, 114
wrap-label	27, 37, 85, 86, 89, 109, 110, 114
wrap-opt	67, 69
keys commands:	
\keys_define:nn	489, 511, 543, 559, 606, 677, 751, 777, 821, 857, 880, 953, 962, 1041, 1058, 1429, 1540, 1783, 1844, 2003, 2045, 2081, 2086, 2400, 2551, 2587, 3025, 3041, 3061, 3081, 4540, 4639, 4755, 4763
\keys_if_exist_p:nn	4751, 4752
\l_keys_key_str	75, 78, 2418, 2573, 3049, 3067, 3089, 4798, 4906
\keys_precompile:nnN	118, 191, 191, 4542, 4546, 4550, 4554, 4558, 4562, 4781
\keys_set:nn	503, 837, 848, 1064, 1434, 1439, 1677, 1682, 1769, 1777, 2438, 3317, 3322, 3517, 4063, 4314, 4594, 4601, 4643, 4648, 4649, 4650, 4651, 4654, 4659, 4660, 4661, 4662, 4663, 4664, 4665, 4697, 4807
\keys_set_known:nn	2705
keyval commands:	
\keyval_parse:NNn	1554, 2111, 4615
L	
label	557, 604, 677
Labels provide by enumext:	
\Alph*	36
\Roman*	36
\alph*	36
\arabic*	30, 36
\roman*	36
\labelsep	100
\labelsep	3675, 3678
labelsep	487
\labelwidth	36, 100
\labelwidth	3675, 3676
labelwidth	487
\leftmargin	90
\leftmargin	89, 3675
legacy commands:	
\legacy_if:nTF	4175, 4178, 4423, 4426
\legacy_if_gset_false:n	375
\legacy_if_set_false:n	4177, 4425
\legacy_if_set_true:n	4137, 4162, 4169, 4182, 4361, 4389, 4430
\linewidth	94
\linewidth	3286, 3374, 3482, 3527, 3702, 3762, 3793, 3915, 3970
\list	359
list-indent	855
list-offset	855
\listparindent	3677
listparindent	855
\lrbox	4193, 4436

M	
\makebox	102
\makebox	2216, 2218, 2992, 4207, 4215, 4219, 4449, 4453
\makelabel	85, 86, 89, 101
\makelabel	85, 88, 3009, 3143
\makesavenoteenv	408
mark-ans	2001
mark-pos	2001, 2043
mark-ref	2001
mini-env	1039
mini-sep	1039
\minipage	365
\miniright	10, 53, 1357, 1401, 1422, 3435, 3580
mode commands:	
\mode_if_math:TF	2481, 2535
\mode_if_vertical:TF	1109, 1137, 1157, 1175, 1274
\mode_leave_vertical:	835, 846, 908, 922, 934, 946, 2214, 2990, 4205
msg commands:	
\msg_error:nn	1403, 1424, 2442, 2475, 2479, 2533, 2641, 3508, 3512, 3601, 3661, 3692, 4048, 4295, 4307, 4666, 4725
\msg_error:nnn	582, 629, 646, 699, 1361, 1368, 1373, 1399, 1421, 1689, 1693, 1808, 2424, 2483, 2501, 2513, 2521, 2525, 2529, 2537, 2579, 3055, 3073, 3095, 4052, 4300, 4528, 4537, 4608, 4713, 4744, 4753, 4790, 4811
\msg_error:nnnn	2427, 2455, 2459, 2463, 2467, 2582, 3058, 3076, 3098, 3499, 3597, 3605, 4589, 4793
\msg_error:nnnnn	530, 550, 2024
\msg_fatal:nn	3305
\msg_fatal:nnn	441
\msg_info:nnn	13, 16, 21, 24, 390, 404
\msg_line_context:	4871, 4876, 4881, 4910, 4915, 4920, 4935, 4950, 4954, 4958, 4962, 4966, 4970, 4977, 4984, 4990, 5004, 5008, 5013, 5017, 5021, 5025, 5030, 5034, 5038, 5042, 5047, 5082, 5086, 5091, 5096, 5100, 5105, 5181, 5185, 5190, 5195, 5200, 5204, 5208, 5212, 5216, 5220, 5224, 5228, 5232
\msg_log:nnn	1828, 1833, 1838
\msg_log:nnnnn	352, 1971, 1976, 1981
\msg_log:nnnnnn	344
\msg_new:nnn	4838, 4842, 4846, 4850, 4855, 4868, 4873, 4878, 4883, 4892, 4900, 4904, 4908, 4913, 4918, 4933, 4948, 4952, 4956, 4960, 4964, 4968, 4972, 4981, 4987, 4993, 4997, 5001, 5006, 5011, 5015, 5019, 5023, 5028, 5032, 5036, 5040, 5045, 5080, 5084, 5089, 5094, 5098, 5103, 5179, 5183, 5188, 5193, 5198, 5202, 5206, 5210, 5214, 5218, 5222, 5226, 5230
\msg_new:nnnn	4859, 5050, 5059, 5068, 5074, 5107, 5117, 5127, 5137, 5147, 5157, 5167, 5173
\msg_term:nnnn	1792, 1797, 3211, 3221, 3252, 3257
\msg_term:nnnnn	1952
\msg_warning:nn	3434, 3579
\msg_warning:nnnn	1989, 1995, 3159, 3164, 3816, 3829, 3865, 3878
\msg_warning:nnnnn	1947, 1957
\multicolsep	94
\multicolsep	1169, 3409, 3552
N	
\NeedsTeXFormat	3
\newcounter	444
\NewDocumentCommand	1357, 2430, 3593, 4512, 4567, 4673, 4722, 4800
\NewDocumentEnvironment	3262, 3459, 3634, 4018, 4268

\newenvsc 2544

\newlabel 35

\newlabel 426

no-store 1842

\noindent 3532, 3924, 3979, 4031, 4248, 4280, 4491

\nointerlineskip 1177, 1180, 3532, 3924, 3979

noitemsep 775

\nopagebreak ... 1120, 1148, 1177, 1180, 1285, 1348, 1354

\normalfont 2389, 2912, 2925, 4409

nosep 775

P

Packages:

caption 105

enumext 25, 35, 38, 62, 90, 98, 123

enumitem 35, 36

expl3 101

footnotehyper 34

hyperref 29, 30, 34, 35, 73, 83, 110, 123

lua-visual-debug 50

multicol 25, 123

scontents 25, 76, 77

shortlst 101

\par .. 1120, 1148, 1180, 1285, 1348, 1354, 1392, 1414, 2366, 3424, 3440, 3444, 3445, 3446, 3569, 3584, 3714, 3942, 3956, 3997, 4011, 4248, 4262, 4491, 4507

\parbox 2008

\parindent 4225, 4459

\parsep 48, 50, 99, 100

\parsep 3243, 3674, 3681, 3686

parsep 775

\parskip 4226, 4460

\partopsep 100

\partopsep 3244, 3679

partopsep 775

peek commands:

 \peek_meaning:NTF 4114, 4128, 4145, 4156, 4338, 4352, 4369

 \peek_meaning_remove:NTF 4121, 4345

 \peek_remove_spaces:n 3131

\phantomsection 35

\phantomsection 415

prg commands:

 \prg_do_nothing: 419

 \prg_new_protected_conditional:Npnn ... 209

 \prg_replicate:nn 226

 \prg_return_false: 213

 \prg_return_true: 212

\printkeyans 16, 117, 4567

prop commands:

 \prop_const_from_keyval:Nn 4714

 \prop_count:N 346, 2140, 2291, 2392, 2802, 2915, 2928, 4412, 4816

 \prop_get:NnNTF 4740

 \prop_gput_if_not_in:Nnn 2138

 \prop_if_exist:NTF 1826, 4532, 4809

 \prop_item:Nn 4534, 4833

 \prop_new:N 1829

\ProvidesExplPackage 4

R

\raggedcolumns 3412, 3558

\ref 71, 82

ref 557, 604, 677

\refstepcounter 4184, 4432

regex commands:

 \regex_match:nnTF .. 211, 726, 728, 740, 742, 2637

 \regex_replace_once:nnN 219

\renewcommand 592, 637, 654, 707

\RenewDocumentCommand 1401, 1422, 2998, 3009, 3127, 3143, 3690, 3724

\RequirePackage 17, 25

resume 1538

resume* 1538

rightmargin 855

\Roman 36, 41

\Roman 463

\roman 36, 41

\roman 464, 575, 4557

S

\s 2638

save-ans 1781

save-key 2079

save-ref 2001

save-sep 2001

scan commands:

 \scan_stop: 100, 3688, 4030, 4279, 4523, 4526

scontents internal commands:

 \l_scontents_fname_out_tl 2597

 __scontents_parse_environment_keys:n . 2603

 __scontents_rescan_tokens:n 2610

 \l_scontents_storing_bool 2595

 \l_scontents_writing_bool 2596

seq commands:

 \seq_clear:N 4675, 4818

 \seq_const_from_clist:Nn 4668

 \seq_count:N 347, 3647, 4679

 \seq_gclear:N 3722, 3723

 \seq_gput_right:Nn 2147, 3735, 3736

 \seq_if_empty:NTF 3741, 4582, 4693

 \seq_if_exist:NTF 1831, 4580

 \seq_if_in:NnTF 4587

 \seq_item:Nn 2635, 3711

 \seq_map_function:NN 4684

 \seq_map_inline:Nn 4595, 4602, 4694, 4695

 \seq_map_pairwise_function:NNN 3743

 \seq_new:N 122, 123, 125, 139, 163, 164, 1834

 \seq_pop_left:NN 4683

 \seq_put_right:Nn 3607, 4691, 4707, 4828

 \seq_set_from_clist:Nn 4676

 \seq_set_map_e:NNn 4685

 \seq_show:N 4584

 \seq_use:Nn 191, 192, 4824

series 1538

\setcounter 737, 741, 743, 3202, 3246, 3652

\setenumext 6, 119, 4673

\setenumextmeta 6, 121, 4714

show-ans 2001, 2043

show-length 951

show-pos 2043

skip commands:

 \skip_add:Nn 1089, 1095, 1101, 1111, 1115, 1139, 1143, 1159, 1192, 1198, 1204, 1223, 1276, 1280, 3674

 \skip_gset:Nn 1296, 1300, 1304

 \skip_gzero_new:N 1291, 1292

 \skip_horizontal:N 923, 935, 947, 4208, 4222, 4456

 \skip_horizontal:n ... 909, 2215, 2223, 2991, 2993, 4206, 4465

<code>\skip_if_eq:nnTF</code>	1087, 1093, 1099, 1190, 1196, 1202, 1216, 1221, 1242, 1293, 1315, 1446, 1460, 1474, 1485, 1496, 1507, 1518, 1529
<code>\skip_new:N</code>	... 81, 82, 83, 87, 88, 89, 90, 91, 143, 183
<code>\skip_set:Nn</code>	1072, 1076, 1125, 1129, 1153, 1218, 1219, 1220, 1227, 1231, 1235, 1244, 1249, 1253, 1256, 1260, 1264, 1295, 1299, 1317, 1321, 1325, 1331, 1335, 1339, 3668, 3682
<code>\skip_set_eq:NN</code>	1164, 1169, 3200, 3242, 3243, 4225, 4226, 4459, 4460
<code>\skip_use:N</code>	1074, 1078, 1113, 1117, 1121, 1141, 1145, 1155, 1161, 1229, 1447, 1451, 1454, 1461, 1465, 1468, 3424, 3444
<code>\skip_vertical:N</code>	... 376, 379
<code>\skip_zero:N</code>	1168, 1181, 3244, 3409, 3552, 3679, 3680
<code>\skip_zero_new:N</code>	1211, 1212, 1213, 1290, 1312, 1313, 1314
<code>\c_zero_skip</code>	. 376, 379, 1087, 1093, 1099, 1190, 1196, 1202, 1216, 1221, 1242, 1293, 1315, 1447, 1461, 1474, 1485, 1496, 1507, 1518, 1529
<code>\small</code>	... 4545, 4549, 4553, 4557, 4561, 4565
<code>\star</code>	... 3029
<code>start</code>	... 749
<code>start*</code>	... 749
<code>\stepcounter</code>	... 3614, 3728
str commands:	
<code>\c_backslash_str</code>	2483, 4871, 4876, 4881, 4886, 4888, 4890, 4895, 4897, 4995, 4999, 5003, 5013, 5017, 5025, 5026, 5030, 5042, 5043, 5047, 5048, 5069, 5071, 5075, 5077, 5105, 5168, 5170, 5174, 5176, 5185, 5186, 5190, 5195, 5196, 5200, 5204, 5208
<code>\c_colon_str</code>	... 2290, 2801, 4523
<code>\c_left_brace_str</code>	... 4976, 4983, 4989
<code>\c_right_brace_str</code>	... 4976, 4983, 4989
<code>\str_case:nn</code>	... 231, 290
<code>\str_case:nnTF</code>	. 1561, 1569, 2118, 2126, 4622, 4631
<code>\str_clear:N</code>	... 3314, 4062
<code>\str_count:n</code>	... 226
<code>\str_if_empty:N</code>	... 1578, 1619, 1646
<code>\str_if_eq:nnTF</code>	... 3203, 3248, 4724
<code>\str_if_in:nnTF</code>	... 4519
<code>\str_new:N</code>	... 129, 178
<code>\str_set:Nn</code>	... 546, 547, 548, 2021, 2022, 2048, 2049
<code>\string</code>	... 408
<code>\strutbox</code>	. 1183, 1192, 1198, 1204, 1218, 1219, 1220, 1223, 1233, 1237, 1246, 1253, 1258, 1266, 1295, 1296, 1299, 1306, 1319, 1327, 1333, 1341, 3440, 3446, 3684

T

TeX and \LaTeX 2_ε commands:

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

tex commands:

<code>\tex_newlinechar:D</code>	... 2609
---------------------------------	----------

text commands:

<code>\text_expand:n</code>	... 4515
<code>\textasteriskcentered</code>	... 2018, 2035
<code>\thepage</code>	... 430

tl commands:

<code>\c_space_tl</code>	2883, 4920, 4935, 4958, 4962, 5149, 5150, 5159, 5160, 5220, 5224
<code>\tl_clear:N</code>	. 516, 522, 1999, 2065, 2075, 2096, 2104, 2310, 2629, 2630, 2744, 2818, 4375
<code>\tl_clear_new:N</code>	... 473

<code>\tl_const:Nn</code>	... 50, 457
<code>\tl_gclear:N</code>	. 338, 339, 340, 1599, 1604, 2719, 3020, 3960, 4015, 4209
<code>\tl_gclear_new:N</code>	... 1586
<code>\tl_gput_right:Nn</code>	... 458
<code>\tl_greplace_all:Nnn</code>	... 479
<code>\tl_gset:Nn</code>	266, 267, 281, 282, 1587, 1600, 1605, 1824, 2633, 2967, 4151
<code>\tl_gset_eq:NN</code>	... 475, 2963, 4202
<code>\tl_if_blank:nTF</code>	2422, 2440, 2577, 3053, 3071, 3093, 4200, 4788
<code>\tl_if_empty:N</code>	. 580, 599, 627, 644, 664, 671, 697, 714, 1612, 1617, 1639, 1644, 1702, 1766, 1774, 1803, 1863, 2154, 2185, 2330, 2674, 2696, 2726, 2755, 2828, 2877, 2988, 4378, 4705
<code>\tl_if_empty:nTF</code>	... 1667
<code>\tl_if_exist:N</code>	... 1672
<code>\tl_if_novalue:nTF</code>	. 2436, 2752, 2826, 2862, 2942, 2961, 2969, 3103, 3312, 3645, 3726, 4060, 4312, 4376
<code>\tl_map_inline:Nn</code>	... 217, 476
<code>\tl_new:N</code>	42, 43, 44, 47, 52, 53, 56, 57, 63, 65, 66, 68, 69, 103, 104, 105, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 124, 126, 127, 128, 130, 133, 134, 146, 154, 155, 156, 159, 177
<code>\tl_put_left::N</code>	... 2663
<code>\tl_put_left:Nn</code>	2162, 2193, 2315, 2657, 2670, 2676, 2686, 2894, 2934, 3945, 4000, 4394, 4397
<code>\tl_put_right:Nn</code>	474, 590, 635, 652, 705, 2166, 2197, 2244, 2254, 2267, 2282, 2288, 2293, 2317, 2322, 2329, 2332, 2342, 2347, 2350, 2356, 2747, 2750, 2757, 2759, 2786, 2791, 2796, 2799, 2808, 2821, 2824, 2830, 2835, 2845, 4380, 4381
<code>\tl_remove_all:Nn</code>	... 4704
<code>\tl_remove_once:Nn</code>	... 2232, 2771
<code>\tl_replace_all:Nnn</code>	... 478, 4739
<code>\tl_reverse:N</code>	... 2231, 2233, 2770, 2772
<code>\tl_set:Nn</code>	. 58, 235, 245, 294, 295, 302, 303, 310, 311, 443, 517, 521, 526, 527, 579, 624, 696, 906, 920, 932, 944, 1701, 1802, 2066, 2076, 2097, 2105, 2386, 2597, 2864, 2909, 2922, 4383, 4406, 4702, 4738, 4808
<code>\tl_set_eq:NN</code>	484, 585, 588, 632, 634, 649, 651, 702, 704, 2230, 2769, 2782, 3115, 3119, 3626, 3628
<code>\tl_to_str:n</code>	... 1672, 1678, 1683, 4515
<code>\tl_trim_spaces:n</code>	... 474, 4691, 4702, 4708, 4724
<code>\tl_use:N</code>	. 480, 483, 601, 666, 673, 716, 977, 981, 985, 989, 993, 997, 1001, 1005, 1009, 1013, 1017, 1021, 1025, 1029, 1033, 1037, 2220, 2237, 2245, 2256, 2269, 2274, 2285, 2950, 2956, 2984, 3011, 3012, 3019, 3106, 3110, 3118, 3145, 3146, 3152, 3269, 3465, 3631, 3952, 4007, 4212, 4223, 4227, 4446, 4457, 4463, 4468, 4570, 4571, 4572, 4573, 4574, 4592, 4687, 4806

token commands:

<code>\token_to_str:N</code>	... 426
<code>topsep</code>	... 775
<code>\topskip</code>	... 1168
<code>\typeout</code>	... 394, 397, 407, 408

U

<code>\u</code>	... 220, 2638
<code>unknown</code>	... 3039, 3061, 3079
use commands:	
<code>\use:N</code>	... 227, 3016, 3271
<code>\use:n</code>	... 1552, 2109, 4521, 4613
<code>\use_none:nn</code>	... 418, 4745
<code>\usecounter</code>	... 3201, 3245

V

\value 1615, 1621, 1628, 1634, 1642, 1648, 1655, 1661

vbox commands:

 \ vbox_set_top:Nn 3950, 4005

\vspace . 836, 847, 1451, 1454, 1465, 1468, 1478, 1480, 1489,
1491, 1500, 1502, 1511, 1513, 1522, 1524, 1533, 1535,
3642, 3653, 4263, 4508

W

widest 749

wrap-ans 2001

wrap-label 487

wrap-label* 487

wrap-opt 2001

Z

\z 2638