

enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-11-08*

©2024 by Pablo González†

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

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

Abstract

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

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	12
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	13
1.3.2	Public dimension	3	6.2	The command <code>\anskey</code>	13
1.3.3	Support for <code>multicol</code>	4	6.2.1	Keys for <code>\anskey</code>	13
1.3.4	Support for <code>minipage</code>	4	6.3	The environment <code>anskey*</code>	14
1.3.5	The <code>\label</code> and <code>\ref</code> system	4	6.3.1	Keys for <code>anskey*</code>	14
1.3.6	Support for <code>\footnote</code>	4	6.4	The environment <code>keyans</code>	15
2	The environments provided	5	6.4.1	The <code>\item*</code> in <code>keyans</code>	15
2.1	The environment <code>enumext</code>	5	6.5	The environment <code>keyanspic</code>	16
2.2	The environment <code>enumext*</code>	5	6.5.1	Keys for <code>keyanspic</code>	16
2.3	The command <code>\item*</code>	5	6.5.2	The command <code>\anspic</code>	16
2.3.1	Keys for <code>\item*</code>	6	6.6	Printing stored content	18
2.4	The command <code>\item</code> in <code>enumext*</code>	6	6.6.1	The command <code>\getkeyans</code>	18
3	The command <code>\setenumext</code>	6	6.6.2	The command <code>\foreachkeyans</code>	18
4	The command <code>\setenumextmeta</code>	6	6.6.3	The command <code>\printkeyans</code>	18
5	The <code>keyval</code> system	7	7	Full examples	19
5.1	Keys for <code>label</code> and <code>ref</code>	7	8	Tagged PDF examples	22
5.2	Keys for spaces	8	9	The way of non-enumerated lists	22
5.2.1	Vertical spaces	8	10	References	24
5.2.2	Horizontal spaces	9	11	Change history	25
5.3	Keys for add code	9	12	Index of Documentation	26
5.4	Keys for <code>start</code> , <code>series</code> and <code>resume</code>	10	13	Implementation	28
5.5	Keys for <code>multicols</code>	10	14	Index of Implementation	145
5.6	Keys for <code>minipage</code>	11			
5.6.1	The command <code>\miniright</code>	11			
5.6.2	The key <code>mini-right</code>	11			

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 2024-11-08.

†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 L^AT_EX: book, report, article and letter on 10pt, 11pt and 12pt.

The minimum requirement is L^AT_EX release 2024-11-01.

1 Introduction

In the L^AT_EX world there are many useful packages and classes for creating “lists of exercises”, “worksheets” or “multiple choice questions”, classes like exam[1] and packages like xsim[2] do the job perfectly, but they don’t always fit the basic day to day needs.

In my work (and in the work of many teachers) it is common to use “simple exercise sheets” also known as “informal lists of exercises”, as an example:

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

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

3. True False

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
- (a) $\alpha > \delta$

(b) L^AT_EXze is cool?

Sometimes we are also interested in showing the “answers” along with the questions:

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

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

3. True False

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
- * $(x - 1)^2$

* $3(x + y + z)$

(a) $\alpha > \delta$

(b) L^AT_EXze is cool?
- * Yes

* Yes, dnf

* doesn’t exist for now :(

* xsim

* very good

* 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

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

2. Second type of questions

3. Third type of questions
- A) value

B) correct

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

II. $\alpha = \delta$

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

A) I only

B) II only

C) I and II only

D) I and III only

E) I, II, and III

D) value

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

5. Question with image on right side:
- A) value

B) value

C) value

D) correct

E) value
- A) value

B) value

C) value

D) correct

E) value

Where what we are interested in the $\langle label \rangle$ and a “short note” that we leave as an explanation, and then print them:

1. B) $x = 5$

2. D)

3. C) some note
- ⌘ 4. E) A duck

⌘ 5. D) “other note”

⌘

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

- 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.

1.1 Description and usage

The `enumext` package defines enumerated environments using the `list` environment provided by \TeX , 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 `luatex enumext.ins` and move all files to appropriate locations, then run `mktexlsr`. To produce the documentation run `arara enumext.dtx`.

<code>enumext.sty</code>	»	<code>TDS:tex/latex/enumext/</code>
<code>enumext.pdf</code>	»	<code>TDS:doc/latex/enumext/</code>
<code>README.md</code>	»	<code>TDS:doc/latex/enumext/</code>
<code>enumext.dtx</code>	»	<code>TDS:source/latex/enumext/</code>
<code>enumext.ins</code>	»	<code>TDS:source/latex/enumext/</code>

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.

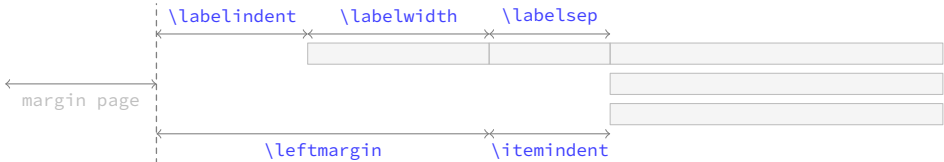


Figure 1: Representation of horizontal lengths in `enumitem`.

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.

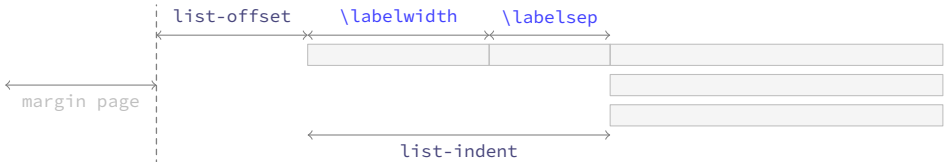


Figure 2: Representation of horizontal lengths concept in `enumext`.

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` and `\foreachkeyans` to print all *stored content*, `\miniright` for `minipage`, `\setenumext` and `\setenumextmeta` to config [*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.

1.3.6 Support for \footnote

The `enumext*` and `keyans*` environments and the `mini-env` key use the `minipage` environment in their implementation but in a transparent way for the user, i.e. it is only used for typesetting and not directly. The `enumext` package provides an *internal implementation* for the command `\footnote` compatible with the `hyperref` package to work in the same way as if it were used anywhere in the document.

Unfortunately, if *tagging* PDF is not enabled, it will not produce the expected “links” because the internal implementation uses `\footnotetext[⟨number⟩]` and `\footnotemark[⟨number⟩]{⟨text⟩}` and support for these is limited by the `hyperref` package.

The best way to solve this if *tagged* PDF is NOT active 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). Load it is as follows:

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

At the moment the `footnotehyper` package is not compatible with *tagged* PDF.

2 The environments provided

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

enumext	<code>\begin{enumext}[⟨keyval list⟩]</code>	<code>\begin{enumext*}[⟨keyval list⟩]</code>
enumext*	<code>\item ⟨item content⟩</code>	<code>\item ⟨item content⟩</code>
	<code>\item [⟨custom⟩] ⟨item content⟩</code>	<code>\item [⟨custom⟩] ⟨item content⟩</code>
	<code>\item* [⟨symbol⟩] [⟨offset⟩] ⟨item content⟩</code>	<code>\item* [⟨symbol⟩] [⟨offset⟩] ⟨item content⟩</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[⟨custom⟩]` 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 `[⟨key = val⟩]` 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 `shortenumerate` or `tasks` environments provided by the `shortlst`[15] and `tasks`[16] packages, `\item` and `\item[⟨custom⟩]` work as usual. The options can be configured globally using `\setenumext` command and locally using `[⟨key = val⟩]` 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 content” 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 (see §1.3.6 for full support).
- You cannot have any standard list environments like `itemize`, `enumerate`, `description`, `quote`, `quotation`, `verse`, `center`, `flushleft`, `flushright`, `verbatim`, `tabbing`, `trivlist`, `list` and all environments created with `\newtheorem`.

Example with `columns=2`

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

2.3 The command \item*

```
\item* \item* [⟨symbol⟩] [⟨offset⟩]
```

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 `⟨offset⟩` set by the the *second optional argument*. 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*` = {<symbol>} default: \textborn
Sets the *symbol* to be displayed in the “left” of the box containing the current <label> set by `labelwidth` key for `\item*` in `enumext` and `enumext*`. The *symbol* can be in *text* or *math* mode, for example `item-sym*={\star}`.

`item-pos*` = {<rigid length>} default: by levels
Sets the *offset* between the box containing the current <label> defined by `labelwidth` key and the <symbol> 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 “first optional argument” `\item(<columns>)` 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{<key = val>}</code>	<code>\setenumext[<keyans*>]{<key = val>}</code>
	<code>\setenumext[<enumext, level>]{<key = val>}</code>	<code>\setenumext[<print, level>]{<key = val>}</code>
	<code>\setenumext[<enumext*>]{<key = val>}</code>	<code>\setenumext[<print, *>]{<key = val>}</code>
	<code>\setenumext[<keyans>]{<key = val>}</code>	<code>\setenumext[<print*>]{<key = val>}</code>

The command `\setenumext` sets the <keys> 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 <keys> set in the *optional argument* 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 {<key name>}{<key-one = val, key-two = val, ...>}</code>
	<code>\setenumextmeta*{<key name>}{<key-one = val, key-two = val, ...>}</code>
	<code>\setenumextmeta [<enumext*>]{<key name>}{<key-one = val, key-two = val, ...>}</code>
	<code>\setenumextmeta [<enumext, level>]{<key name>}{<key-one = val, key-two = val, ...>}</code>

The command `\setenumextmeta` adds a new “meta-key” for the environments `enumext` and `enumext*`, the {<key name>} 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 argument* ‘*’ will create the new “meta-key” for the environment `enumext*` and for all levels of the environment `enumext`. For example: `\setenumextmeta*{midsep}{topsep=3pt, partopsep=0pt}` will create a new key `midsep` available for all levels of the `enumext` environment and the `enumext*` environment and we can use it like any other key so `\begin{enumext}[midsep]` and `\begin{enumext*}[midsep]` will be valid.

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

`mode-box` $\langle value forbidden \rangle$ default: *not used*

This is a “*switch-key*” that does not receive an argument and is “*only*” available for the “*first level*” of the `enumext` environment and the `enumext*` environment. When this is set the `label`, `font`, `wrap-label` and `wrap-label*` keys are executed within `\makebox` for the `enumext` and `keyans` environments.

- This key is intended for compatibility with *tagged* PDF and is forcibly “*enabled*” when `\DocumentMetadata` is present. If you want to get the same document output whether `\DocumentMetadata` is active or not, you must enable this key.

- In the `enumext*` and `keyans*` environments `\makeLabel` are redefined using `\makebox` by default. If `enumext` or `keyans` is used in the `enumext*` environment the key must be activated manually.

`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 label and ref*” system with the `save-ref` key. You cannot use commands with $\langle label \rangle$ as an argument, for example `\emph{\backslash alph*}` will return an error. For full customization of how $\langle label \rangle$ is displayed use the `font`, `wrap-label` and/or `wrap-label*` keys.

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

Wraps the *current* $\langle label \rangle$ defined by `label` key referenced by $\{ \#1 \}$ after executing the `align` and `font` keys. The $\{ \langle code \rangle \}$ must be passed between braces and this 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 \mywrap { s m }
{
  \IfBooleanTF{\#1}
  {
    {\textcolor{red}{\textbf{Q}}\textcolor{blue}{\textbf{.}}\textcolor{gray}{\#2}}
    {\textcolor{blue}{\textbf{Q}}\textcolor{red}{\textbf{.}}\textcolor{gray}{\#2}}
  }
}
```

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

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

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

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

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

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

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

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`.

- In the `enumext*` and `keyans*` environments this value is passed to `\\parskip` within the `minipage` environment where “item content” is placed.

`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. T_EX 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`.

- In the `enumext*` and `keyans*` environments this value corresponds to the separation between rows.

`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 “switch-key” that does not receive an argument available *only* for the “first level” of environment `enumext`. Fix the *baseline* when an environment `enumext` is nested in `enumext*` 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`.

- This key is provided as a way to work around this minor issue, but you should be aware that if for some reason you have the `itemindent` key set in the `enumext*` environment it will be lost and you will need to adjust it using the `list-offset` key in the `enumext` environment.

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

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

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

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

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

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

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

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

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

5.2.2 Horizontal spaces

`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 environments `enumext` and `keyans` 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”.

- The `enumext*` and `keyans*` environments are implemented using `\makebox` and `minipage` which causes “list indent” to always be equal to the value passed to `labewidth` plus `labelsep`. Passing a value to this key is equivalent to setting the value for the `list-offset` key.

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

Sets the extra *horizontal indentation*, beyond `labelsep`, of the “first line” off each `\item` that is not followed by a “blank line” or the `\par` command. This value must be greater than or equal to `0pt` and is applied internally using `\hspace` without modifying the value of `\itemindent`.

- This key is intended for the `enumext*` and `keyans*` environments where, by their implementation, it is not possible to adjust `labelwidth` and `list-indent` without modifying the output. If you use `enumext` or `keyans` and want to get around the *blank line* limitation or the `\par` command followed by `\item` you can modify `labelwidth` and `list-indent` and get the same effect.

`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.

- In the `enumext*` and `keyans*` environments this value is passed to `\parindent` within the `minipage` environment where “item content” is placed.

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*` = {`<code>`} default: *not used*
 Execute {`<code>`} “before” the environment starts. The {`<code>`} must be passed between braces, is executed “before” performing all calculations related to the `list parameters` and [`<key = val>`] sets in the environment that is, before the arguments defining the environment are executed: {`<code>`}\begin{list}{`<arg one>`}{`<arg two>`}.

`first` = {`<code>`} default: *not used*
 Executes {`<code>`} when “starting” the environment. The {`<code>`} 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}{`<arg one>`}{`<arg two>`}{`<code>`}\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.
- 🔴 In the `enumext*` and `keyans*` environments this key is executed after the `listparindent`, `parsep` and `itemindent` keys within the `minipage` environment in which the “item content” is placed.

`after` = {`<code>`} default: *not used*
 Execute {`<code>`} “after” finishing the environment. The {`<code>`} must be passed between braces.

5.4 Keys for start, series and resume

`start` = {`<integer | integer expression>`} default: `1`
 Sets the *start value* of the numbering on the current level. The {`<integer expression>`} 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*` = {`<integer | string>`} default: *not used*
 Sets the *start value* of the numbering on the current level. Internally `<string>` 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 multicol

`columns` = {`<integer>`} default: `1`
 Set the *number of columns* to be used by the `multicol` environment within the environment. The value must be a positive integer less than or equal to `10`.

`columns-sep` = {`<rigid length>`} default: *by level*
 Set the *space between columns* used by the `multicol` environment within the environment. Internally sets the value of `\columnsep`, by default its value is equal to the sum of the values set in the keys `labelwidth` and `labelsep` of the current level.

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”.

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 `⟨key⟩` is “*active*” the `\anskey` command and the environments `anskey*`, `keyans`, `keyans*` and `keyanspic` are available.

<pre>\begin{enumext}[save-ans={⟨store name⟩}] \item Text \anskey{answer} \item Text \begin{keyans} ... \end{keyans} \end{enumext}</pre>	<pre>\begin{enumext}[save-ans={⟨store name⟩}] \item Text \anskey{answer} \item Text \begin{keyanspic} ... \end{keyanspic} \end{enumext}</pre>
---	---

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

For security reasons the *optional argument* of the inner levels or the nested environment are *filtered* by excluding all `⟨keys⟩` related to the “*storage system*” (§6.1) along with the keys `mini-env`, `mini-sep`, `mini-right`, `mini-right*`, `series`, `resume` and `resume*` when storing in *sequence* `{⟨store name⟩}` set by `save-ans` key.

6.1 Keys for storage system

The only `⟨keys⟩` available for all levels of the `enumext` environment and the `enumext*` environment are `no-store` and `save-key`, the rest of the `⟨keys⟩` 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 *sequence* and *prop list* in which the `{⟨contents⟩}` will be “*stored*” by `\anskey` and `anskey*` in `enumext` and `enumext*` environments and the current `⟨labels⟩` for `\item*` and `\anspic*` in the environments `keyans`, `keyans*` and `keyanspic`. If the *sequence* or *prop list* `{⟨store name⟩}` 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 argument* of the inner levels or nested environment that will be passed to the *sequence*. The `⟨key list⟩` passed to this key ignores any `⟨keys⟩` in the “*stored structure*” 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 “*stored keys*” by default in the *sequence* $\{\langle store name \rangle\}$ would be `nosep`, `columns=2`, but using the key `save-key={columns=3}` will overwrite and the “*stored key*” in the *sequence* $\{\langle store name \rangle\}$ are only `columns=3` ignoring all the others.

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

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

`no-store` $\langle value forbidden \rangle$ default: *not used*

This is a “*switch-key*” that does not receive an argument and disables the “*storing content*” in the *sequence* and *prop list* $\{\langle store name \rangle\}$ 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` command or use `anskey*` environment and “*without*” interfering with the `check-ans` key.

6.1.1 Keys for label and ref

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

Activates the “*internal label and ref*” mechanism for referencing “*stored content*” in *prop list* $\{\langle store name \rangle\}$ 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 *prop list* $\{\langle store name \rangle\}$ 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` in *prop list* `test` within the environment in which the key `save-ans=test` was set.

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

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

`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`. The “*symbol*” is placed in a box of width equal to the value of `labelwidth` at the current level, separated by the value of the key `mark-sep` and aligned by the value of the key `mark-pos`. This key is not affected by the keys `font` or `wrap-label` so if you want to apply *styling* you have to do it directly, for example: `mark-ans={\textcolor{red}{\textbf{\textasteriskcentered}}}`

`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 `mark-sep` key.

`mark-sep` = $\{\langle rigid length \rangle\}$ default: `\labelsep`

Sets the *horizontal space* between the box containing the “*symbol*” defined by `mark-ans` key and the current $\langle label \rangle$ for `\item*` and `\anspic*` in the `keyans`, `keyans*` and `keyanspic` environments, the *argument* passed to the `\anskey` and the *body* in `anskey*` environment.

`wrap-ans` = $\{\langle code \{ \#1 \} more code \rangle\}$ 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 how the *argument* or *body* is displayed 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-key` = $\{\langle code \{ \#1 \} more code \rangle\}$ default: *not used*

Wraps the *current* $\langle label \rangle$ when using the `show-ans` key for `\item*` and `\anspic*` referenced by $\{\#1\}$ in the `keyans`, `keyans*` and `keyanspic` environments after executing the `align` and `font` keys. The $\{\langle code \rangle\}$ must be passed between braces and *only* affects how the $\langle label \rangle$ is displayed and NOT the “*stored label*” in the *sequence* and *prop list* $\{\langle store name \rangle\}$ set by `save-ans` key. This key overwrites the key `wrap-label` and if is passed using `\setenumext` it is necessary to use double ‘ $\{\#\#1\}$ ’. For example, if you want the $\langle label \rangle$ to be displayed in red when using `show-ans` you just set `wrap-key={\textcolor{red}{\#1}}`.

`wrap-opt` = {`<code #1 more code>`} default: `[[#1]]`
 Wraps the *optional argument* passed to the `\item*` and `\anspic*` referenced by `{#1}` in the `keyans`, `keyans*` and `keyanspic` environments when using the `show-ans` or `show-pos` keys. The `{<code>}` must be passed between braces and only affects the current *optional argument* and NOT the “stored content” in the *sequence* and *prop list* `{<store name>}` set by `save-ans` key. If this key is passed using `\setenumext` it is necessary to use double `{##1}`.

`show-ans` = {`<true | false>`} default: `false`
 Displays the *argument* passed to the `\anskey`, the *body* for `anskey*` environment, the `<label>` for `\item*` and `\anspic*` at the place where it is executed. If the *optional argument* is present in `\item*` or `\anspic*` it will be shown using `wrap-opt` key.

6.1.3 Keys for debug and checking

`show-pos` = {`<true | false>`} default: `false`
 Displays the *position* occupied by the “stored content” by `\anskey`, `anskey*`, `\item*` and `\anspic*` in the *prop list* `{<store name>}` set by `save-ans` key. This position is used by the `\getkeyans` command and by the `\ref` command if the `save-ref` key is active.

`check-ans` = {`<true | false>`} default: `false`
 Enables the *checking answer* mechanism displaying an appropriate message on the terminal. This key works under the logic that each `\item` or `\item*` that does not open an inner level or nested environment contains “only one answer” or “only one execution” of the `\anskey` or `anskey*`. It is intended to be used in conjunction with the `no-store` key.

6.2 The command `\anskey`

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

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

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

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

6.2.1 Keys for `\anskey`

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

`break-col` `<value forbidden>` default: `not used`
 Stores `{<content>}` in the *sequence* `{<store name>}` of the form `\columnbreak \item <content>`.

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

`item-star` `<value forbidden>` default: `not used`
 Stores `{<content>}` in the *sequence* `{<store name>}` of the form `\item* <content>`.

`item-sym*` = {`<symbol>`} default: `not set`
 Sets the *symbol* for `\item*` when using the key `item-star` and stores `{<content>}` in the *sequence* `{<store name>}` of the form `\item* [<symbol>] <content>`. The *symbol* can be in text or math mode, for example `item-sym*={\ast}` stores `\item*[\ast] <content>`.

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

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.

*

2. Text containing our instructions or questions.

(a) Question.

*

3. Text containing our instructions or questions.

*

4. Text containing our instructions or questions.

*

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* `\{store name\}` 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 L^AT_EX 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]
      \langle first answer \rangle
    \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.

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

```
\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*`. The keys `mark-ans`, `mark-pos`, `mark-sep`, `save-sep`, `wrap-opt`, `wrap-key`, `show-ans` and `show-pos` are available for both environments.

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 *optional argument* `\langle content \rangle` 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.

Example

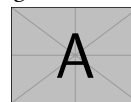
```
\begin{enumext}[save-ans=test,columns=2,show-ans=true]
  \item Text containing a question.

  \begin{keyans*}[nosep,columns=2]
    \item Choice
    \item* Correct choice
    \item Choice
    \item Choice
    \item Choice
  \end{keyans*}

  \item Text containing a question and image.

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

1. Text containing a question.
 - A) Choice
 - C) Choice
 - E) 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 key = val \rangle] \anspic*[\langle content \rangle]{\langle drawing or tabular \rangle} \end{keyanspic}
```

The `keyanspic` environment is an “*enumerated list*” environment activated by the `save-ans` key that has the same configuration for “*spacing*” and `\label` as the `keyans` environment that uses the `\anspic` command instead of `\item`. It is intended for placing *drawings or tabular* with `\label` centered *above* or *below* in a *single line* or *upper and lower* layout style.

When the `keyanspic` environment is used *without keys* the `\label`s are centered *below* the *drawings or tabular* in a *single line* layout style.

A representation of the output can be seen in the figure 6.

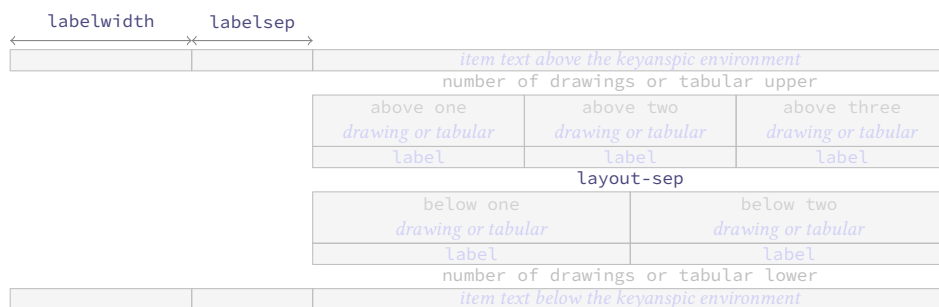


Figure 6: Representation of the `keyanspic` environment with `layout-sty={\langle 3, 2 \rangle}` in `enumext`.

This environment cannot be nested and must always be at the “*first level*” of the `enumext` environment, the `\item` command is disabled and keys cannot be set using `\setenumext`.

6.5.1 Keys for keyanspic

`label-pos = {\langle above | below \rangle}` default: *below*

Set the *position* of `\label` to be centered “*above*” or “*below*” *drawings or tabular* when the `\anspic` command is executed.

`label-sep = {\langle rubber length | rigid length \rangle}` default: *internal adjustment*

Set the *vertical spacing* between the `\label` centered “*above*” or “*below*” and *drawings or tabular* when running the `\anspic` command.

`layout-sty = {\langle n° upper , n° lower \rangle}` default: *not set*

Set the *number of drawings or tabular* that will be distributed “*upper*” and “*lower*” within the environment when executing the `\anspic` command. The value must be passed in braces and if not set or the `\langle n° lower \rangle` is omitted the *drawings or tabular* will be put on a *single line*.

`layout-sep = {\langle rubber length | rigid length \rangle}` default: *adjusted parsep from keyans*

Set the *vertical separation* between the number of *drawings or tabular* placed at the “*upper*” and “*lower*” within the environment when executing the `\anspic` command. Internally adjusts the `parsep` value taken from the `keyans` environment.

`layout-top = {\langle rubber length | rigid length \rangle}` default: *adjusted topsep from keyans*

Set the *vertical space* added to both the top and bottom of the environment. Internally adjust the value of `topsep` taken from `keyans` environment.

The keys `mark-ans`, `mark-pos`, `mark-sep`, `save-sep`, `wrap-opt`, `wrap-key`, `show-ans` and `show-pos` are available for this environment.

6.5.2 The command \anspic

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

The `\anspic` command take three arguments, the *starred argument* ‘`*`’ store the current `\label` next to the *optional argument* `\langle content \rangle` 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 and labels below.

  \begin{keyanspic}[layout-sty={3,2}]
    \anspic{\includegraphics[scale=0.15]{example-image-a}}
    \anspic{\includegraphics[scale=0.15]{example-image-b}}
    \anspic{\includegraphics[scale=0.15]{example-image-a}}
    \anspic{\includegraphics[scale=0.15]{example-image-a}}
    \anspic*[note]{\includegraphics[scale=0.15]{example-image-a}}
  \end{keyanspic}

  \item Question with images and labels above.

  \begin{keyanspic}[label-pos=above, layout-sty={3,2},layout-sep=0.25cm]
    \anspic{\includegraphics[scale=0.15]{example-image-a}}
    \anspic{\includegraphics[scale=0.15]{example-image-b}}
    \anspic{\includegraphics[scale=0.15]{example-image-a}}
    \anspic{\includegraphics[scale=0.15]{example-image-a}}
    \anspic*[note]{\includegraphics[scale=0.15]{example-image-a}}
  \end{keyanspic}

  \item Question with images and labels below on a single line.

  \begin{keyanspic}
    \anspic{\includegraphics[scale=0.15]{example-image-a}}
    \anspic{\includegraphics[scale=0.15]{example-image-b}}
    \anspic{\includegraphics[scale=0.15]{example-image-a}}
    \anspic{\includegraphics[scale=0.15]{example-image-a}}
    \anspic*[note]{\includegraphics[scale=0.15]{example-image-a}}
  \end{keyanspic}
\end{enumext}
```

1. Question with images and labels below.

A

A)

B

B)

A

C)

A

D)

A

* E) [note]

2. Question with images and labels above.

A)

A

B)

B

C)

A

D)

A

* E) [note]

A

3. Question with images and labels below on a single line.

A

A)

B

B)

A

C)

A

D)

A

* E) [note]

Remember to pass the `alt={description}` key to the `\includegraphics` command when creating a tagged PDF.

6.6 Printing stored content

6.6.1 The command `\getkeyans`

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

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

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

6.6.2 The command `\foreachkeyans`

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

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

Options for command

`sep = {<code>}` default: {;}

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

`step = {<integer>}` default: 1

Sets the *step* (increment) applied to the value set by key `start` for “each” `{<content>}` 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 will finish. 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>{<store name>}
\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` or `enumext*` environment if the *starred argument* ‘*’ is used.

The “stored content” can only be accessed *after* it is stored in the *sequence*, if `{<store name>}` does not exist the command will return an error.

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

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

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

- If we execute `\printkeyans*{<store name>}` and the *sequence* `{<store name>}` already contains any `enumext*` environment an error will be returned as we cannot nest.

- If we execute `\printkeyans*{⟨store name⟩}` and the *sequence* $\{⟨store name⟩\}$ contains any `enumext` environments, they will start with the $\langle keys \rangle$ set for the first level unless they are set in the *optional argument* or `save-key` is used to modify it.
- If we execute `\printkeyans{⟨store name⟩}` and the *sequence* $\{⟨store name⟩\}$ contains any environment `enumext*`, they will start with the $\langle keys \rangle$ set by default unless they are set in the *optional argument* or `save-key` is used to modify it.

The default values for the “first level” of `\printkeyans` commands and `\printkeyans*` are established using `\setenumext[⟨print , 1⟩]{⟨keys⟩}` and `\setenumext[⟨print*⟩]{⟨keys⟩}`.

If we need to set the $\langle keys \rangle$ for the environment `enumext` “saved” in the *sequence* $\{⟨store name⟩\}$ we will use `\setenumext[⟨print , level⟩]{⟨keys⟩}` and if we need to set the $\langle keys \rangle$ for the environment `enumext*` “saved” in the *sequence* $\{⟨store name⟩\}$ we will use `\setenumext[⟨print , *⟩]{⟨keys⟩}`.

Example

```
\begin{enumext}[save-ans=sample,columns=1,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) ~~LaTeXe~~ is cool?
[2]

3. Related to Linux
(a) You use linux?
[3]

(b) Rate the following package and class
i. `xsim`
[4]

ii. `exsheets`
[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.

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

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


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

2. A

3. B

4. A

Example 2

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

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

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

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

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

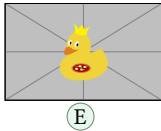
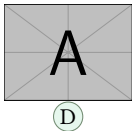
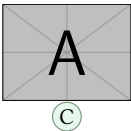
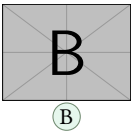
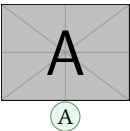
 value

D

 value

E

 value
4. Question with image and label below:



5. Question with image on right 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) ~~ET~~Xze 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

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 

<div>1</div> <div>Which choice best describes what happens in the passage?</div> <div>A) One character argues with another character who intrudes on her home.</div> <div>B) One character receives a surprising request from another character.</div> <div>C) One character reminisces about choices she has made over the years.</div> <div>D) One character criticizes another character for pursuing an unexpected course of action.</div>	<div>3</div> <div>Which choice best describes what happens in the passage?</div> <div>A) One character argues with another character who intrudes on her home.</div> <div>B) One character receives a surprising request from another character.</div> <div>C) One character reminisces about choices she has made over the years.</div> <div>D) One character criticizes another character for pursuing an unexpected course of action.</div>
<div>2</div> <div>Which choice best describes what happens in the passage?</div> <div>A) One character argues with another character who intrudes on her home.</div> <div>B) One character receives a surprising request from another character.</div> <div>C) One character reminisces about choices she has made over the years.</div> <div>D) One character criticizes another character for pursuing an unexpected course of action.</div>	<div>4</div> <div>Which choice best describes what happens in the passage?</div> <div>A) One character argues with another character who intrudes on her home.</div> <div>B) One character receives a surprising request from another character.</div> <div>C) One character reminisces about choices she has made over the years.</div> <div>D) One character criticizes another character for pursuing an unexpected course of action.</div>
1. A)	3. B)
2. C)	4. D)













8 Tagged PDF examples

This section is just to show the compatibility of `enumext` with *tagged* PDF using `lualatex`. The attached files here are just for testing and are intended as examples and, in a way, to simplify the time of Matthew Bertucci (@mbertucci) when he sees this excellent package and adds it to [The LaTeX Tagged PDF repository](#).

To compile the tests with `lualatex-dev` the packages `multicol`, `scontents`, `unicode-math`, `geometry`, `graphicx`, `luamml` and `hyperref` are required along with the line:

```
\DocumentMetadata
{
  lang = en-US, pdfversion = 2.0, pdfstandard = ua-2,
  testphase = {phase-III, math, title, table, firstaid},
}
```

♦ All examples have been checked using `veraPDF` together with `ngpdf`.

- The file `enumext-01.tex` contains the basic tests for the `enumext` and `enumext*` environments and the nesting between them plus the use of the `label`, `labelwidth`, `labelsep`, `ref`, `align` and `wrap-label` keys. Source file  and *tagged* PDF .
- The file `enumext-02.tex` contains the tests for the `enumext` and `enumext*` environments and the support for `minipage` and `multicols` environments using the keys `columns`, `columns-sep`, `mini-env`, `mini-right` and `\miniright` command. Source file  and *tagged* PDF .
- The file `enumext-03.tex` contains the tests for the `enumext` and `keyanspic` environments activated by the `save-ans` key together with the `save-sep` and `save-ref` keys and the `\printkeyans` command. Source file  and *tagged* PDF .
- The file `enumext-04.tex` contains the tests for the `\anskey` command and the `anskey*` environment activated by the `save-ans` key along with the `\getkeyans` and `\printkeyans` commands. Source file  and *tagged* PDF .
- The file `enumext-05.tex` contains the tests for the environments `keyans`, `keyans*` and `keyanspic` activated by the key `save-ans` together with the keys `no-store` and `show-ans` and the commands `\setenumext`, `\setenumextmeta`, `\printkeyans` and `\foreachkeyans`. Source file  and *tagged* PDF .
- The file `enumext-06.tex` contains the tests for the environments `enumext` and `enumext*` for *fake* `itemize` and `description`. Source file  and *tagged* PDF .

9 The way of non-enumerated lists

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

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

Fake itemize environment

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

- | | |
|---------------------|---------------------|
| • First level item | * First level item |
| – Second level item | ◇ Second level item |
| • Third level item | ◦ Third level item |
| · Fourth level item | ★ Fourth level item |
| • First level item | * First level item |

Fake description environment

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

Something A short one-line description.

This is an entry *without* a label.

Something A short *one-line* description text.

Something long A much *longer* description text may take more than one line or more than one paragraph.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

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

Something A short one-line description.

This is an entry *without* a label.

Something A short *one-line* description text.
Something long A much *longer* description text may take more than one line or more than one paragraph. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

- The small space at the beginning of the “*unlabeled entry*” corresponds to `\labelsep` and can be removed using `\hspace{-\labelsep}` at the beginning of the line.
- When *tagged* PDF is active the default `description` style is NOT available due to the redefinition of `\makeLabel` for the `align` key which uses `\makebox` in this case, meaning that `\item[⟨content⟩]` will not extend beyond `\labelwidth` which causes overlaps,

Description indented by label

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

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

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

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

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

Something A short one-line description.
This is an entry *without* a label.
Something A short one-line description.
Something long A much longer description. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.
If we add `align=right` it will look like this:

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

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

Description with multi-line labels

The `label` key does not accept *multiline material*, this is where the `wrap-label` and `wrap-label*` keys 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 }  
{%  
  \SuspendTagging{\parbox}%  
  \IfBooleanTF{#1}  
  {%  
    {\strut\smash{\parbox[t]{\labelwidth}{\raggedright{#2}}}}%  
    {\strut\smash{\parbox[t]{\labelwidth}{\raggedleft{#2}}}}%  
  }\ResumeTagging{\parbox}%  
}
```

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

Final notes

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

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

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

Why list environments?

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

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

Why not random questions and other utilities

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

Why has it taken so long?

One of the setbacks, beyond my laziness, was including compatibility with *tagged* PDF. To be honest, it’s something I never considered at any point, but I firmly believe that being able to create *accessible documents* provides a great opportunity in the world of mathematics education. From my perspective as a *high school* teacher, beyond theorems and deep mathematics, the use of exercise lists is one of the most common things. Being able to open the way to work in parallel with those who have different abilities is really important and I regret not having looked into this in the past. I hope that `enumext` serves this purpose and inspires more users and authors to follow this path.

10 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>, 2024.
- [5] The \LaTeX Project. “enumerate – Enumerate with redefinable labels”. Available from CTAN, <https://www.ctan.org/pkg/enumerate>, 2024.
- [6] BEZOS, JAVIER. “Customizing lists with the enumitem package”. Available from CTAN, <https://www.ctan.org/pkg/enumitem>, 2019.
- [7] BERRY, KARL. “ $\text{\LaTeX}_{2\epsilon}$: An Unofficial Reference Manual”. Available from CTAN, <https://ctan.org/pkg/latex2e-help-texinfo>, 2024.
- [8] The \LaTeX Project. “Extensive support for hypertext in \LaTeX ”. Available from CTAN, <https://www.ctan.org/pkg/hyperref>, 2024.
- [9] BURNOL, JEAN-FRANÇOIS. “The footnotehyper package”. Available from CTAN, <https://www.ctan.org/pkg/footnotehyper>, 2021.
- [10] The \LaTeX Project. “The expl3 package”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2024.
- [11] The \LaTeX Project. “The \LaTeX_{X3} 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.
- [17] FISCHER, ULRIKE. “tagpdf – \LaTeX kernel code for PDF tagging”. Available from CTAN, <https://www.ctan.org/pkg/tagpdf>, 2024.
- [18] The \LaTeX Project. “latex-lab – \LaTeX laboratory”. Available from CTAN, <https://www.ctan.org/pkg/latex-lab>, 2024.
- [19] MITTELBACH, FRANK. “ \LaTeX ’s socket management”. Available from CTAN, <https://ctan.org/tex-archive/macros/latex/base>, 2024.

11 Change history

v1.0 2024-11-08 – First public release.

12 Index of Documentation

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

C		F	
Document class:		\footnote	5
article	2	I	
book	2	\itemsep	8
exam	2	K	
letter	2	Keys for \anskey provide by enumext:	
report	2	break-col	13
\columnbreak	4, 13	item-join	13
\columnsep	10	item-pos*	13
Commands provide by enumext:		item-star	13
\anskey	11-14	item-sym*	13
\anspic	11-13, 16	Keys for \foreachkeyans provide by enumext:	
\foreachkeyans	18	after	18
\getkeyans	13, 18	before	18
\item*	5-7, 11-13, 15	sep	18
\item	5-7, 10, 11, 13, 15, 16	start	18
\miniright	11	step	18
\printkeyans	6, 12, 18	stop	18
\setenumextmeta	6	wrapper	18
\setenumext	5-7, 11-13, 15, 19	Keys for anskey* provide by enumext:	
Counters defined by enumext:		break-col	13
enumXiii	4	force-eol	14
enumXii	4	item-join	13
enumXiv	4	item-pos*	13
enumXi	4	item-star	13
enumXviii	4	item-sym*	13
enumXvii	4	overwrite	14
enumXvi	4	write-env	14
enumXv	4	Keys for environments provide by enumext:	
E		above*	9
Environments provide by enumext:		above	8, 9
anskey*	11-14, 22	after	10
enumext*	4-15, 18, 19, 22	align	7, 12, 22, 23
enumext	4-16, 18, 19, 22	base-fix	8
keyans*	4-15, 22	before*	9, 10
keyanspic	4, 7, 8, 11-14, 16, 22	before	9
keyans	4-16, 22	below*	9
Environments:		below	9
Verbatim	14	check-ans	12, 13
center	5	columns-sep	4, 10, 22
description	5, 22, 23	columns	4, 9, 10, 22
enumerate	1, 3, 5, 24	first	10
figure	5	font	7, 12
flushleft	5	item-pos*	5, 6
flushright	5	item-sym*	5, 6
itemize	5, 22	itemindent	8-10
list	3, 5, 9, 24	itemsep	8
minipage	3-5, 8-11, 22, 24	label-pos	16
multicols	3, 4, 10, 22	label-sep	16
quotation	5	labelsep	3-7, 9, 10, 22, 23
quote	5	labelwidth	3, 4, 6, 7, 9, 10, 12, 22, 23
shortenenumerate	5	labelwith	5
tabbing	5	label	7, 8, 10, 15, 22, 23
table	5	labewdith	9
tasks	5	layout-sep	16
trivlist	5	layout-sty	16
verbatim	5	layout-top	16
verse	5	list-indent	3, 9
		list-offset	3, 8, 9, 23

listparindent	9, 10	Labels provide by enumext:	
mark-ans	12, 15, 16	\Alph*	7, 8, 15
mark-pos	12, 15, 16	\Roman*	7, 8
mark-ref	12	\alph*	7, 8
mark-sep	12, 15, 16	\arabic*	7, 8
mini-env	4, 9, 11, 22	\roman*	7, 8
mini-right*	7, 11	\labelsep	3, 7
mini-right	7, 11, 22	\labelwidth	3, 7
mini-sep	4, 11	\linewidth	11
mode-box	7	\listparindent	9
no-store	11–14, 22		
noitemsep	8	P	
nosep	8, 22	Packages:	
overwrite	14	enumerate	24
parsep	8, 10, 16	enumext	1–5, 7, 16, 22, 24
partopsep	8	enumitem	3, 4, 23, 24
ref	4, 8, 22	fancyvrb	14
resume*	7, 10, 11	footnotehyper	5
resume	7, 10, 11	geometry	22
rightmargin	9	graphicx	22
save-ans	4, 6, 10–16, 18, 22	hyperref	4, 5, 12–14, 22, 24
save-key	10–12, 19	l3keys	7
save-ref	4, 7, 12–14, 18, 22	l3prop	24
save-sep	12, 15, 16, 22	l3seq	24
series	7, 10, 11	luamml	22
show-ans	12, 13, 15, 16, 22	multicol	1, 2, 4, 22, 24
show-length	8	scontents	1, 2, 14, 22
show-pos	12, 13, 15, 16, 18	shortlst	5
start*	10	tasks	5
start	10	task	6
topsep	8, 9, 16	unicode-math	22
widest	7	xsim	2
wrap-ans	12	\parsep	8
wrap-key	12, 15, 16	\partopsep	8
wrap-label*	7, 23		
wrap-label	7, 12, 22, 23	R	
wrap-opt	13, 15, 16	\raggedcolumns	4
write-env	14	\ref	4
		\rightmargin	9
L			
\label	4	T	
		\topsep	8

13 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.

13.1 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.

13.2 Initial set up

Start the DocStrip guards.

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

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

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

13.3 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}[2024-11-01]
```

Now declare the `enumext` package.

```
4 \ProvidesExplPackage {enumext} {2024-11-08} {1.0} {Enumerate exercise sheets}
```

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

```
5 \hook_gput_code:nnn {begindocument} {enumext}
6 {
7   \IfPackageLoadedTF { multicol }
8   {
9     \msg_info:nnn { enumext } { package-load } { multicol }
10  }
11  {
12    \msg_info:nnn { enumext } { package-not-load } { multicol }
13    \RequirePackage{multicol}[2024-05-23]
14  }
15  \IfPackageLoadedTF { scontents }
16  {
17    \msg_info:nnn { enumext } { package-load } { scontents }
18  }
19  {
20    \msg_info:nnn { enumext } { package-not-load } { scontents }
21    \RequirePackage{scontents}
22  }
23 }
```

13.4 Definition of variables

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

```
\__enumext_level_int
\__enumext_level_h_int
\__enumext_anskey_level_int
\__enumext_keyans_level_int
\__enumext_keyans_level_h_int
\__enumext_keyans_pic_level_int
```

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

```
24 \int_new:N \__enumext_level_int
25 \int_new:N \__enumext_level_h_int
26 \int_new:N \__enumext_anskey_level_int
27 \int_new:N \__enumext_keyans_level_int
28 \int_new:N \__enumext_keyans_level_h_int
29 \int_new:N \__enumext_keyans_pic_level_int
```

(End of definition for `__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:` (§13.5.1).

```

30 \bool_new:N \l__enumext_starred_bool
31 \bool_new:N \g__enumext_starred_bool
32 \bool_new:N \l__enumext_starred_first_bool
33 \bool_new:N \l__enumext_standar_bool
34 \bool_new:N \g__enumext_standar_bool
35 \bool_new:N \l__enumext_standar_first_bool
36 \bool_new:N \l__enumext_anskey_env_bool
37 \bool_new:N \l__enumext_keyans_env_bool
38 \tl_new:N \g__enumext_start_line_tl
39 \tl_new:N \g__enumext_envir_name_tl
40 \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` (§13.11) and then modified by the function `__enumext_label_style:Nnn` used by `label` key (§13.14).

```

41 \cs_set_protected:Npn \__enumext_tmp:n #1
42 {
43   \tl_new:c { l__enumext_counter_#1_tl }
44 }
45 \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 (§13.14).

```

46 \tl_const:Nn \c__enumext_counter_style_tl
47 { { { arabic } { roman } { Roman } { alph } { Alph } } }
48 \tl_new:N \l__enumext_ref_key_arg_tl
49 \tl_new:N \l__enumext_ref_the_count_tl
50 \cs_set_protected:Npn \__enumext_tmp:n #1
51 {
52   \tl_new:c { l__enumext_renew_the_count_#1_tl }
53   \tl_new:c { l__enumext_the_counter_#1_tl }
54   \tl_set:ce { l__enumext_the_counter_#1_tl } { \exp_not:c { theenumX#1 } }
55 }
56 \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 (§13.25).

```

57 \int_new:N \g__enumext_resume_int
58 \int_new:N \g__enumext_resume_vii_int
59 \tl_new:N \l__enumext_resume_name_tl
60 \bool_new:N \l__enumext_resume_active_bool
61 \tl_new:N \g__enumext_standar_series_tl
62 \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` (§13.15) and `label` (§13.13) keys.

```

63 \dim_new:N \l__enumext_current_widest_dim
64 \tl_new:N \g__enumext_counter_styles_tl
65 \tl_new:N \g__enumext_widest_label_tl
66 \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 (§13.18). The variables `\l__enumext_leftmargin_X_dim` and `\l__enumext_itemindent_X_dim` are used and set by the function `__enumext_calc_hspace:NNNNNNNNNN` (§13.38.1).

```
67 \cs_set_protected:Npn \__enumext_tmp:n #1
68 {
69   \bool_new:c { \l__enumext_leftmargin_tmp_#1_bool }
70   \dim_new:c { \l__enumext_leftmargin_tmp_#1_dim }
71   \dim_new:c { \l__enumext_leftmargin_#1_dim }
72   \dim_new:c { \l__enumext_itemindent_#1_dim }
73 }
74 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }
```

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

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

Internal variables used by `columns` key (§13.22) and `align` key (§13.13).

```
75 \cs_set_protected:Npn \__enumext_tmp:n #1
76 {
77   \skip_new:c { \l__enumext_multicols_above_#1_skip }
78   \skip_new:c { \l__enumext_multicols_below_#1_skip }
79   \skip_new:c { \g__enumext_multicols_right_#1_skip }
80   \str_new:c { \l__enumext_align_label_pos_#1_str }
81 }
82 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }
```

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

```
\g__enumext_minipage_stat_int
\l__enumext_minipage_temp_skip
\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 (§13.23.4) and the keys `mini-right`, `mini-right*`, `mini-env` and `mini-sep` (§13.21, §13.23).

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

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

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

The 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 (§13.13), the integer `\l__enumext_start_X_int` are used by the `start` and `start*` keys (§13.15), the token list `\l__enumext_fake_item_indent_X_tl` is used by `itemindent` key (§13.18.1), the variables `\l__enumext_label_fill_left_X_tl` and `\l__enumext_label_fill_right_X_tl` are used by the `align` key (§13.13). 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 (§13.20).

```
96 \cs_set_protected:Npn \__enumext_tmp:n #1
97 {
98   \bool_new:c { \l__enumext_wrap_label_#1_bool }
99   \bool_new:c { \l__enumext_wrap_label_opt_#1_bool }
100   \int_new:c { \l__enumext_start_#1_int }
101   \tl_new:c { \l__enumext_fake_item_indent_#1_tl }
102   \tl_new:c { \l__enumext_label_fill_left_#1_tl }
103   \tl_new:c { \l__enumext_label_fill_right_#1_tl }
104   \bool_new:c { \l__enumext_vspace_a_star_#1_bool }
105   \bool_new:c { \l__enumext_vspace_b_star_#1_bool }
106 }
107 \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }
```

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

```

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

```

The variable `\l__enumext_store_active_bool` setting by `save-ans` key (§13.26.1) activates all the mechanism related to `\anskey`, `anskey*`, `keyans`, `keyans*` and `keyanspic` environments.

The variable `\l__enumext_store_name_tl` saves the $\{\langle store\ name\rangle\}$ 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 $\{\langle store\ name\rangle\}$ used by different functions.

The variable `\l__enumext_store_anskey_arg_tl` save the *argument* of `\anskey` (§13.30) and the variables `\l__enumext_store_anskey_env_tl` and `\l__enumext_store_anskey_opt_tl` save the $\langle body\rangle$ and the $\langle keys\rangle$ of the environment `anskey*` (§13.31).

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*` (§13.37) and `\anspic*` (§13.42.2) for the `keyans`, `keyans*` and `keyanspic` environments.

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

```

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

```

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

```

\l__enumext_setkey_tmpa_tl
\l__enumext_setkey_tmpb_tl
\l__enumext_setkey_tmpa_int
\l__enumext_setkey_tmpa_seq
\l__enumext_setkey_tmpb_seq

```

Internal variables used by the command `\setenumext` (§13.48).

```

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

```

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

```

\l__enumext_meta_path_tl
\l__enumext_foreach_print_seq
\l__enumext_foreach_name_prop_tl
\g__enumext_foreach_default_keys_tl

```

Internal variables used by the `\printkeyans` command (§13.47) and `\foreachkeyans` command (§13.50).

```

122 \tl_new:N \l__enumext_meta_path_tl
123 \seq_new:N \l__enumext_foreach_print_seq
124 \tl_new:N \l__enumext_foreach_name_prop_tl
125 \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_print_keyans_star_bool
\l__enumext_mark_position_str
\l__enumext_mark_sep_tmpa_dim
\l__enumext_mark_sep_tmpb_dim
\l__enumext_show_pos_tmp_int
\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` (§13.47), `show-pos`, `show-ans`, `mark-pos`, `mark-sep` keys (§13.27), `item-sym*` key (§13.35), `save-key` key (§13.27.2) and “*storing structure*”.

```

126 \tl_new:N \l__enumext_print_keyans_starred_tl
127 \bool_new:N \l__enumext_print_keyans_star_bool
128 \str_new:N \l__enumext_mark_position_str
129 \dim_new:N \l__enumext_mark_sep_tmpa_dim
130 \dim_new:N \l__enumext_mark_sep_tmpb_dim
131 \int_new:N \l__enumext_show_pos_tmp_int
132 \tl_new:N \g__enumext_item_symbol_aux_tl
133 \cs_set_protected:Npn \__enumext_tmp:n #1
134 {
135   \tl_new:c { \l__enumext_print_keyans_#1_tl }
136   \tl_new:c { \l__enumext_store_save_key_#1_tl }
137   \bool_new:c { \l__enumext_store_save_key_#1_bool }
138   \bool_new:c { \l__enumext_store_upper_level_#1_bool }
139 }
140 \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_anspic_args_seq
\l__enumext_anspic_mini_width_dim
\l__enumext_anspic_above_int
\l__enumext_anspic_below_int
\l__enumext_anspic_label_above_bool
\l__enumext_anspic_mini_pos_str
\l__enumext_anspic_label_box
\l__enumext_anspic_body_box

```

Internal variables used by `keyanspic` environment and `\anspic` command (§13.42.1).

```

141 \seq_new:N \l__enumext_anspic_args_seq
142 \dim_new:N \l__enumext_anspic_mini_width_dim
143 \int_new:N \l__enumext_anspic_above_int
144 \int_new:N \l__enumext_anspic_below_int
145 \bool_new:N \l__enumext_anspic_label_above_bool
146 \str_new:N \l__enumext_anspic_mini_pos_str

```

```

147 \box_new:N \l__enumext_anspic_label_box
148 \box_new:N \l__enumext_anspic_body_box
149 \dim_new:N \l__enumext_anspic_label_htdp_dim
150 \dim_new:N \l__enumext_anspic_body_htdp_dim

```

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

Internal variables used by “*internal check answer*” mechanism (§13.26.3) used by the `check-ans`, `no-store`, `wrap-key` keys and check for starred commands `\item*` in `keyans` and `keyans*` environments and `\anspic*` in `keyanspic` environment.

```

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

```

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

The boolean variable `\l__enumext_hyperref_bool` will determine if the `hyperref` package is present or load in memory (§13.7). The boolean variable `\l__enumext_footnotes_key_bool` determine if `hyperref` is load with key `hyperfootnotes=true`.

```

160 \bool_new:N \l__enumext_hyperref_bool
161 \bool_new:N \l__enumext_footnotes_key_bool

```

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

Internal variables used by `save-ref` key (§13.27). The variables `\l__enumext_label_copy_X_tl` correspond to temporary copies of the $\langle labels \rangle$ defined by level on which operations will be performed.

The variables `\l__enumext_newlabel_arg_one_tl` and `\l__enumext_newlabel_arg_two_tl` will be used to form the arguments passed to the function `__enumext_newlabel:nn` (§13.7) and the variable `\l__enumext_write_aux_file_tl` will be in charge of executing the writing code in the `.aux` file.

```

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

```

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

Internal variables used for redefinition of `\footnote` (§13.8).

```

170 \int_new:N \g__enumext_footnote_standar_int
171 \int_new:N \g__enumext_footnote_starred_int
172 \seq_new:N \g__enumext_footnote_standar_arg_seq
173 \seq_new:N \g__enumext_footnote_starred_arg_seq
174 \seq_new:N \g__enumext_footnote_standar_int_seq
175 \seq_new:N \g__enumext_footnote_starred_int_seq

```

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

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

```

176 \cs_set_protected:Npn \__enumext_tmp:n #1
177 {
178   \bool_new:c { l__enumext_item_starred_#1_bool }
179   \int_new:c { l__enumext_item_column_pos_#1_int }
180   \int_new:c { g__enumext_item_count_all_#1_int }
181   \int_new:c { l__enumext_joined_item_#1_int }
182   \int_new:c { l__enumext_joined_item_aux_#1_int }
183   \int_new:c { l__enumext_tmpa_#1_int }
184   \dim_new:c { l__enumext_tmpa_#1_dim }
185   \box_new:c { l__enumext_item_text_#1_box }
186   \dim_new:c { l__enumext_joined_width_#1_dim }
187   \dim_new:c { l__enumext_item_width_#1_dim }
188   \tl_new:c { g__enumext_item_symbol_aux_#1_tl }

```

```

189 \str_new:c { \__enumext_align_label_#1_str }
190 \bool_new:c { g__enumext_minipage_active_#1_bool }
191 \box_new:c { \__enumext_miniright_code_#1_box }
192 \bool_new:c { g__enumext_minipage_center_#1_bool }
193 \dim_new:c { g__enumext_minipage_right_#1_dim }
194 \skip_new:c { g__enumext_minipage_right_#1_skip }
195 }
196 \clist_map_inline:nn { vii, viii } { \__enumext_tmp:n {#1} }

```

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

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

```

197 \clist_const:Nn \c__enumext_all_envs_clist
198 {
199   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
200   {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
201 }

```

(End of definition for `\c__enumext_all_envs_clist`.)

13.5 Some utility functions

`\keys_precompile:neN` `\seq_use:NV` Non-standard kernel variants used by the `\printkeyans` command (§13.47) and `\foreachkeyans` command (§13.50).

```

202 \cs_generate_variant:Nn \keys_precompile:nnN { neN }
203 \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.

```

204 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
205 {
206   \hook_gput_code:nnn {begindocument} {enumext} { #1 }
207 }

```

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

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

```

208 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
209 {
210   \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
211 }
212 \cs_new_protected:Npn \__enumext_before_env:nn #1 #2
213 {
214   \hook_gput_code:nnn {env/#1/before} {enumext} {#2}
215 }

```

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

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

```

216 \cs_new:Nn \__enumext_level:
217 {
218   \int_to_roman:n { \__enumext_level_int }
219 }

```

(End of definition for `__enumext_level:`.)

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

```

220 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
221 {
222   \regex_match:nnTF { ^[\+|-]?[\d]+$ } {#1} % $
223   { \prg_return_true: }
224   { \prg_return_false: }
225 }

```

(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.

```

226 \cs_new_protected:Nn \__enumext_regex_counter_style:
227 {
228   \tl_map_inline:Nn \c__enumext_counter_style_tl
229   {
230     \regex_replace_once:nnN { \c{##1}\* }
231     { \c{##1}\cB{\u{\__enumext_ref_the_count_tl}\cE} } \l__enumext_ref_key_arg_tl
232   }
233 }

```

(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.

```

234 \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
235 {
236   * ~ #2
237   \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
238   = ~ \use:c { #1_use:c } { l__enumext_#2_#3_#1 } \\
239 }

```

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

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

```

240 \cs_new_protected:Nn \__enumext_unskip_unkern:
241 {
242   \int_case:nnT { \lastnodetype }
243   {
244     { 11 }{ \unskip }
245     { 12 }{ \unkern }
246   }
247 }

```

(End of definition for `__enumext_unskip_unkern:.`)

13.5.1 Utilities for environments and levels

`__enumext_is_not_nested:` The function `__enumext_is_not_nested:` set the variables `\g__enumext_standar_bool` and `\g__enumext_starred_bool` to “true” only if the environments `enumext` and `enumext*` are NOT nested in each other and save the environment name in `\l__enumext_envir_name_tl`.

```

248 \cs_new_protected:Nn \__enumext_is_not_nested:
249 {
250   \str_case:en { \@currentenv }
251   {
252     {enumext}
253     {
254       \tl_set:Nn \l__enumext_envir_name_tl { enumext }
255       \bool_lazy_and:nnT
256       { \bool_not_p:n { \g__enumext_standar_bool } }
257       { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
258       {
259         \bool_gset_true:N \g__enumext_standar_bool
260       }
261     }
262     {enumext*}
263     {
264       \tl_set:Nn \l__enumext_envir_name_tl { enumext* }
265       \bool_lazy_and:nnT
266       { \bool_not_p:n { \g__enumext_starred_bool } }
267       { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
268       {
269         \bool_gset_true:N \g__enumext_starred_bool
270       }
271     }
272   }
273 }

```


The function `__enumext_is_on_first_level:` will set the variables `\l__enumext_standar_first_bool` (§13.26.1), `\l__enumext_starred_first_bool` (§13.26.1) and `\l__enumext_anskey_env_bool` (§13.31) 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.

```

274 \cs_new_protected:Nn \__enumext_is_on_first_level:
275 {
276   \bool_lazy_all:nT
277   {
278     { \bool_if_p:N \g__enumext_standar_bool }
279     { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
280     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
281   }
282   {
283     \bool_set_true:N \l__enumext_standar_first_bool
284     \bool_set_true:N \l__enumext_anskey_env_bool
285     \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
286     \tl_gset:Ne \g__enumext_start_line_tl
287     {
288       on ~ line ~ \exp_not:V \inputlineno
289     }
290   }
291   \bool_lazy_all:nT
292   {
293     { \bool_if_p:N \g__enumext_starred_bool }
294     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
295     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
296   }
297   {
298     \bool_set_true:N \l__enumext_starred_first_bool
299     \bool_set_true:N \l__enumext_anskey_env_bool
300     \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
301     \tl_gset:Ne \g__enumext_start_line_tl
302     {
303       on ~ line ~ \exp_not:V \inputlineno
304     }
305   }
306 }

```

(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.

```

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

```

```

332         in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
333     }
334 }
335 }
336 }

```

(End of definition for `__enumext_keyans_name_and_start:`.)

13.5.2 Utilities for log and terminal

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

```

337 \cs_new_protected:Nn \__enumext_reset_global_vars:
338 {
339     \__enumext_reset_global_int:
340     \__enumext_reset_global_bool:
341     \__enumext_reset_global_tl:
342 }
343 \cs_new_protected:Nn \__enumext_reset_global_int:
344 {
345     \int_gzero:N \g__enumext_item_number_int
346     \int_gzero:N \g__enumext_item_anskey_int
347     \int_gzero:N \g__enumext_item_answer_diff_int
348 }
349 \cs_new_protected:Nn \__enumext_reset_global_bool:
350 {
351     \bool_gset_false:N \g__enumext_check_ans_key_bool
352     \bool_gset_false:N \g__enumext_standar_bool
353     \bool_gset_false:N \g__enumext_starred_bool
354 }
355 \cs_new_protected:Nn \__enumext_reset_global_tl:
356 {
357     \tl_gclear:N \g__enumext_store_name_tl
358     \tl_gclear:N \g__enumext_start_line_tl
359     \tl_gclear:N \g__enumext_envir_name_tl
360 }

```

(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.

```

361 \cs_new_protected:Nn \__enumext_log_global_vars:
362 {
363     \msg_log:nneeee { enumext } { prop-seq-int-hook }
364     { \g__enumext_store_name_tl }
365     { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
366     { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
367     { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
368 }

```

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.

```

369 \cs_new_protected:Nn \__enumext_log_answer_vars:
370 {
371     \msg_log:nneeee { enumext } { item-answer-hook }
372     { \int_use:N \g__enumext_item_number_int }
373     { \int_use:N \g__enumext_item_anskey_int }
374     { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int } }
375 }

```

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

13.6 Copying list and minipage environments

The `list` environment provided by \TeX has the following plain form:

```

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

```

And `minipage` environment provided by \TeX 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 `\list` environment or a related command.

- For compatibility with *tagged* PDF we should use `\NewCommandCopy` and not `\cs_new_eq:NN` for `\item`. When *tagged* PDF is active `\item` is redefined using `\ltxcmd` (see `latex-lab-block[18]`).

```

\__enumext_start_list:nn
\__enumext_stop_list:
\__enumext_item_std:w
\__enumext_minipage:w
\__enumext_endminipage:

```

The functions `__enumext_start_list:nn` and `__enumext_stop_list:` correspond to copies of `\list` and `\endlist` from plain definition of `\list` environment, the function `__enumext_item_std:w` is a copy of the `\item` command.

```

376 \__enumext_at_begin_document:n
377 {
378   \cs_new_eq:NN \__enumext_start_list:nn \list
379   \cs_new_eq:NN \__enumext_stop_list: \endlist
380   \NewCommandCopy \__enumext_item_std:w \item
381 }

```

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

```

382 \__enumext_at_begin_document:n
383 {
384   \cs_new_eq:NN \__enumext_minipage:w \minipage
385   \cs_new_eq:NN \__enumext_endminipage: \endminipage
386 }

```

(End of definition for `__enumext_start_list:nn` and others.)

13.7 Compatibility with hyperref and footnotehyper

```

\__enumext_after_hyperref:
\__enumext_hypertarget:nn
\__enumext_phantomsection:

```

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

```

387 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
388 \hook_gset_rule:nnnn { begindocument } { enumext } { after } { hyperref }

```

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

```

389 \cs_new_protected:Nn \__enumext_after_hyperref:
390 {
391   \IfPackageLoadedTF { hyperref }
392   {
393     \msg_info:nnn { enumext } { package-load } { hyperref }
394     \bool_set_true:N \l__enumext_hyperref_bool
395     \IfHyperBoolean{hyperfootnotes}
396     {
397       \bool_set_true:N \l__enumext_footnotes_key_bool
398     }
399     { }
400   }
401   { }

```

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`.

```

402 \bool_if:NT \l__enumext_footnotes_key_bool
403 {
404   \IfPackageLoadedTF { footnotehyper }
405   {
406     \msg_info:nnn { enumext } { package-load } { footnotehyper }
407   }
408   {
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 `\l__enumext_hyperref_bool` is false the functions `__enumext_hypertarget:nn` and `__enumext_phantomsection:` will be disabled.

```

412 \bool_if:NTF \l__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: `\l__enumext_newlabel_arg_one_tl`

#2: `\l__enumext_newlabel_arg_two_tl`

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

```

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 \l__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`.)

13.8 Internal redefining `\footnote` command

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

`__enumext_footnotetext:nn` `__enumext_renew_footnote:` `__enumext_print_footnote:` `__enumext_renew_footnote_mini:` `__enumext_print_footnote_mini:`

Redefinition of the `\footnote` command using `\footnotetext` and `\footnotemark` for the `mini-env` key in the `enumext` and `keyans` environments.

```

437 \cs_new_protected:Nn \__enumext_footnotetext:nn
438 {
439     \footnotetext[#1]{#2}
440 }
441 \cs_new_protected:Nn \__enumext_renew_footnote:
442 {
443     \RenewDocumentCommand \footnote { o +m }
444     {
445         \tl_if_novalue:nTF {##1}
446         {
447             \stepcounter{footnote}
448             \int_gset_eq:Nc \g__enumext_footnote_standar_int { c@footnote }
449         }
450         {
451             \int_gset:Nn \g__enumext_footnote_standar_int { ##1 }
452         }
453         \footnotemark [ \g__enumext_footnote_standar_int ]
454         \seq_gput_right:Nn \g__enumext_footnote_standar_arg_seq { ##2 }
455         \seq_gput_right:NV
456         \g__enumext_footnote_standar_int_seq \g__enumext_footnote_standar_int
457     }
458 }
459 \cs_new_protected:Nn \__enumext_print_footnote:
460 {
461     \seq_if_empty:NF \g__enumext_footnote_standar_int_seq
462     {
463         \seq_map_pairwise_function:NNN

```

```

464         \g__enumext_footnote_standar_int_seq
465         \g__enumext_footnote_standar_arg_seq
466         \__enumext_footnotetext:nn
467     }
468     \seq_gclear:N \g__enumext_footnote_standar_arg_seq
469     \seq_gclear:N \g__enumext_footnote_standar_int_seq
470 }

```

The `enumext*` and `keyans*` environments are implemented using `minipage` so we must also redefine `\footnote` to keep these numbering as if it were part of the document.

```

471 \cs_new_protected:Nn \__enumext_renew_footnote_mini:
472 {
473     \RenewDocumentCommand \footnote { o +m }
474     {
475         \tl_if_novalue:nTF {##1}
476         {
477             \stepcounter{footnote}
478             \int_gset_eq:Nc \g__enumext_footnote_starred_int { c@footnote }
479         }
480         {
481             \int_gset:Nn \g__enumext_footnote_starred_int { ##1 }
482         }
483         \footnotemark [ \g__enumext_footnote_starred_int ]
484         \seq_gput_right:Nn \g__enumext_footnote_starred_arg_seq { ##2 }
485         \seq_gput_right:NV
486             \g__enumext_footnote_starred_int_seq \g__enumext_footnote_starred_int
487     }
488 }
489 \cs_new_protected:Nn \__enumext_print_footnote_mini:
490 {
491     \seq_if_empty:NF \g__enumext_footnote_starred_int_seq
492     {
493         \seq_map_pairwise_function:NNN
494             \g__enumext_footnote_starred_int_seq
495             \g__enumext_footnote_starred_arg_seq
496             \__enumext_footnotetext:nn
497     }
498     \seq_gclear:N \g__enumext_footnote_starred_arg_seq
499     \seq_gclear:N \g__enumext_footnote_starred_int_seq
500 }

```

(End of definition for `__enumext_footnotetext:nn` and others.)

```

\__enumext_renew_footnote_standar:
\__enumext_print_footnote_standar:
\__enumext_renew_footnote_starred:
\__enumext_print_footnote_starred:

```

We encapsulate the redefinition of `\footnote` to pass it to internal `__enumext_mini_page` environment used by the `mini-env` key in the `enumext` and `keyans` environments. We will run the redefinition when `tagged` PDF is active or when the `footnotehyper` package is not loaded.

```

501 \cs_new_protected:Nn \__enumext_renew_footnote_standar:
502 {
503     \bool_if:NT \g__enumext_standar_bool
504     {
505         \IfDocumentMetadataTF
506         {
507             \__enumext_renew_footnote:
508         }
509         {
510             \bool_if:NF \l__enumext_footnotes_key_bool
511             {
512                 \__enumext_renew_footnote:
513             }
514         }
515     }
516 }
517 \cs_new_protected:Nn \__enumext_print_footnote_standar:
518 {
519     \bool_if:NT \g__enumext_standar_bool
520     {
521         \IfDocumentMetadataTF
522         {
523             \__enumext_print_footnote:
524         }
525         {

```

```

526         \bool_if:NF \l__enumext_footnotes_key_bool
527         {
528             \__enumext_print_footnote:
529         }
530     }
531 }
532 }

```

We encapsulate the redefinition of `\footnote` to pass it to the `enumext*` and `keyans*` environments. We will run the redefinition when *tagged* PDF is active or when the `footnotehyper` package is not loaded.

```

533 \cs_new_protected:Nn \__enumext_renew_footnote_starred:
534 {
535     \IfDocumentMetadataTF
536     {
537         \__enumext_renew_footnote_mini:
538     }
539     {
540         \bool_if:NF \l__enumext_footnotes_key_bool
541         {
542             \__enumext_renew_footnote_mini:
543         }
544     }
545 }
546 \cs_new_protected:Nn \__enumext_print_footnote_starred:
547 {
548     \IfDocumentMetadataTF
549     {
550         \__enumext_print_footnote_mini:
551     }
552     {
553         \bool_if:NF \l__enumext_footnotes_key_bool
554         {
555             \__enumext_print_footnote_mini:
556         }
557     }
558 }

```

In `enumext*` and `keyans*` environments we need to use “hooks” to print `\footnote` with support for *tagged* PDF.

```

559 \__enumext_after_env:nn { enumext* }
560 {
561     \__enumext_print_footnote_starred:
562 }
563 \__enumext_after_env:nn { keyans* }
564 {
565     \__enumext_print_footnote_starred:
566 }

```

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

13.9 The internal minipage environment

```

\__enumext_internal_mini_page:
__enumext_mini_env*

```

The function `__enumext_internal_mini_page:` creates a internal `__enumext_mini_page` environment (custom version of `minipage`) setting the `\if@minipage` switch to “false” to allow spaces at the “above” of the environment, plus we will add `\skip_vertical:N \c_zero_skip` to maintain alignment on “top” in the first part and `\skip_vertical:N \c_zero_skip` in the second part to allow spaces “below”. This environment will be used internally by the `mini-env` key, it is NOT documented in the user interface and is for internal use only. Within this environment we redefine `\footnote` to make them look the same as if they were elsewhere in the document. This function is passed to the function `__enumext_safe_exec:` in the `enumext` environment definition (§13.39) and `__enumext_safe_exec_vii:` in the `enumext*` environment definition (§13.44)

```

567 \cs_new_protected:Nn \__enumext_internal_mini_page:
568 {
569     \int_compare:nNtT { \l__enumext_level_int } = { 0 }
570     {
571         \DeclareDocumentEnvironment{__enumext_mini_page}{ m }
572         {
573             \__enumext_renew_footnote_standar:
574             \__enumext_minipage:w [ t ] { ##1 }
575             \legacy_if_gset_false:n { @minipage }
576             \skip_vertical:N \c_zero_skip
577         }
578     }
579 }

```



```

578         {
579             \skip_vertical:N \c_zero_skip
580             \__enumext_endminipage:
581             \__enumext_print_footnote_standar:
582         }
583     }
584 }

```

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

13.10 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.

```

585 \dim_zero_new:N \itemwidth

```

13.11 Definition of counters

```

\__enumext_define_counters:Nn
enumXi
enumXii
enumXiii
enumXiv
enumXv
enumXvi
enumXvii
enumXviii

```

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.

```

586 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
587 {
588     \cs_if_exist:cTF { c@ #2 }
589     { \msg_fatal:nnn { enumext } { counters } { #2 } }
590     {
591         \tl_set:Nn #1 { #2 }
592         \newcounter { #2 }
593     }
594 }

```

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

```

595 \__enumext_define_counters:Nn \__enumext_counter_i_tl { enumXi }
596 \__enumext_define_counters:Nn \__enumext_counter_ii_tl { enumXii }
597 \__enumext_define_counters:Nn \__enumext_counter_iii_tl { enumXiii }
598 \__enumext_define_counters:Nn \__enumext_counter_iv_tl { enumXiv }
599 \__enumext_define_counters:Nn \__enumext_counter_v_tl { enumXv }
600 \__enumext_define_counters:Nn \__enumext_counter_vi_tl { enumXvi }
601 \__enumext_define_counters:Nn \__enumext_counter_vii_tl { enumXvii }
602 \__enumext_define_counters:Nn \__enumext_counter_viii_tl { enumXviii }

```

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

13.12 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`, `enumext*`, `keyans` and `keyans*` environments, so it is necessary to get a default value for `labelwidth` from these `<labels>` at the same time.

```

603 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
604 {
605     \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
606     \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
607 }
608 \__enumext_register_counter_style:Nn \arabic { 0 }
609 \__enumext_register_counter_style:Nn \Alph { M }
610 \__enumext_register_counter_style:Nn \alph { m }
611 \__enumext_register_counter_style:Nn \Roman { VIII }
612 \__enumext_register_counter_style:Nn \roman { viii }

```

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

```

\__enumext_label_width_by_box:Nn
\__enumext_label_width_by_box:cv

```

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

```

613 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
614 {
615   \hbox_set:Nn \l__enumext_label_width_by_box {#2}
616   \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
617 }
618 \cs_generate_variant:Nn \__enumext_label_width_by_box:Nn { cv }

```

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

```

\__enumext_label_style:Nnn
\__enumext_label_style:cvn

```

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

```

619 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
620 {
621   \tl_clear_new:N #1
622   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
623   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
624   \tl_map_inline:Nn \g__enumext_counter_styles_tl
625   {
626     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
627     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
628     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
629   }
630   \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
631   { \tl_use:N \g__enumext_widest_label_tl }
632   \tl_set_eq:cN { the #2 } #1
633 }
634 \cs_generate_variant:Nn \__enumext_label_style:Nnn { cvn }

```

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

13.13 Setting keys associated with label

When `tagged PDF` is active `\makelabel` is redefined using `\makebox` to work correctly (§13.34). From the user side it is convenient to have a key that allows using this redefinition with `\makebox` without having `\IfDocumentMetadataTF` active.

mode-box

We define the key `mode-box` only for the “first level” of `enumext` and `enumext*` environments.

```

635 \cs_set_protected:Npn \__enumext_tmp:n #1
636 {
637   \keys_define:nn { enumext / #1 }
638   {
639     mode-box .bool_set:N = \l__enumext_mode_box_bool,
640     mode-box .initial:n = false,
641     mode-box .value_forbidden:n = true,
642   }
643 }
644 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

(End of definition for `mode-box`.)

font
labelsep
labelwidth
wrap-label
wrap-label*

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

```

645 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
646 {
647   \keys_define:nn { enumext / #1 }
648   {
649     font .tl_set:c = { \l__enumext_label_font_style_#2_tl },
650     font .value_required:n = true,
651     labelsep .dim_set:c = { \l__enumext_labelsep_#2_dim },
652     labelsep .initial:n = {0.3333em},
653     labelsep .value_required:n = true,
654     labelwidth .dim_set:c = { \l__enumext_labelwidth_#2_dim },
655     labelwidth .value_required:n = true,
656     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
657     wrap-label .initial:n = {##1},
658     wrap-label .value_required:n = true,

```

```

659     wrap-label* .code:n = {
660         \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
661         \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
662     },
663     wrap-label* .value_required:n = true,
664 }
665 }
666 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

(End of definition for font and others.)

`align` The `align` key is implemented differently for “starred” and “non starred” environments. For compatibility with tagged PDF we must set `\l__enumext_align_label_pos_X_str`.

```

667 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
668 {
669     \keys_define:nn { enumext / #1 }
670     {
671         align .choice:,
672         align / left .code:n =
673             {
674                 \tl_clear:c { l__enumext_label_fill_left_#2_tl }
675                 \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
676                 \str_set:cn { l__enumext_align_label_pos_#2_str } { l }
677             },
678         align / right .code:n =
679             {
680                 \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
681                 \tl_clear:c { l__enumext_label_fill_right_#2_tl }
682                 \str_set:cn { l__enumext_align_label_pos_#2_str } { r }
683             },
684         align / center .code:n =
685             {
686                 \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
687                 \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
688                 \str_set:cn { l__enumext_align_label_pos_#2_str } { c }
689             },
690         align / unknown .code:n =
691             \msg_error:nneee { enumext } { unknown-choice }
692             { align } { left, ~ right, ~ center } { \exp_not:n {##1} },
693         align .initial:n = left,
694         align .value_required:n = true,
695     }
696 }
697 \clist_map_inline:nn
698 {
699     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
700 }
701 { \__enumext_tmp:nn #1 }
702 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
703 {
704     \keys_define:nn { enumext / #1 }
705     {
706         align .choice:,
707         align / left .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
708         align / right .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
709         align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
710         align / unknown .code:n =
711             \msg_error:nneee { enumext } { unknown-choice }
712             { align } { left, ~ right, ~ center } { \exp_not:n {##1} },
713         align .initial:n = left,
714         align .value_required:n = true,
715     }
716 }
717 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

(End of definition for align.)

13.14 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.

13.14.1 Define and set label and ref keys for enumext environment

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

```

ref
\l__enumext_label_i_tl 718 \cs_set_protected:Npn \l__enumext_tmp:nnn #1 #2 #3
\l__enumext_label_ii_tl 719 {
\l__enumext_label_iii_tl 720 \keys_define:nn { enumext / #1 }
\l__enumext_label_iv_tl 721 {
722     label .code:n = {
723         \__enumext_label_style:cvn { l__enumext_label_#2_tl }
724         { l__enumext_counter_#2_tl } {##1}
725         \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
726         \l__enumext_current_widest_dim
727     },
728     label .initial:n = #3,
729     label .value_required:n = true,
730     ref .code:n = \__enumext_standar_ref:n {##1},
731     ref .value_required:n = true,
732 }
733 }
734 \__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
735 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*) }
736 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
737 \__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.

```

738 \cs_new_protected:Npn \__enumext_standar_ref:n #1
739 {
740     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
741     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
742     {
743         \msg_error:nnn { enumext } { key-ref-empty } { enumext }
744     }
745     {
746         \tl_set_eq:Nc
747         \l__enumext_ref_the_count_tl { l__enumext_counter_ \__enumext_level: _tl }
748         \__enumext_regex_counter_style:
749         \tl_set_eq:Nc
750         \l__enumext_ref_the_count_tl { l__enumext_the_counter_ \__enumext_level: _tl }
751         \tl_put_right:ce { l__enumext_renew_the_count_ \__enumext_level: _tl }
752         {
753             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V \l__enumext_ref_the_count_tl }
754         }
755     }
756 }

```

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

```

757 \cs_new_protected:Nn \__enumext_standar_ref:
758 {
759     \tl_if_empty:cF { l__enumext_renew_the_count_ \__enumext_level: _tl }
760     {
761         \tl_use:c { l__enumext_renew_the_count_ \__enumext_level: _tl }
762     }
763 }

```

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

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

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

```

ref
\l__enumext_label_vii_tl 764 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
\l__enumext_label_viii_tl 765 {
766     \keys_define:nn { enumext / #1 }

```

```

767     {
768         label .code:n = {
769             \__enumext_label_style:cvn { \__enumext_label_#2_tl }
770             { \__enumext_counter_#2_tl } {##1}
771             \dim_set_eq:cN { \__enumext_labelwidth_#2_dim }
772             \__enumext_current_widest_dim
773         },
774         label .initial:n = #3,
775         label .value_required:n = true,
776         ref .code:n = \__enumext_starred_ref:n {##1},
777         ref .value_required:n = true,
778     }
779 }
780 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
781 \__enumext_tmp:nnn { keyans* } { viii } { \Alph*.}

```

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

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

```

\__enumext_starred_ref:
782 \cs_new_protected:Npn \__enumext_starred_ref:n #1
783 {
784     \tl_set:Nn \__enumext_ref_key_arg_tl {#1}
785     \int_compare:nNt { \__enumext_level_h_int } = { 1 }
786     {
787         \tl_if_empty:NTF \__enumext_ref_key_arg_tl
788         {
789             \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
790         }
791         {
792             \tl_set_eq:NN \__enumext_ref_the_count_tl \__enumext_counter_vii_tl
793             \__enumext_regex_counter_style:
794             \tl_set_eq:NN \__enumext_ref_the_count_tl \__enumext_the_counter_vii_tl
795             \tl_put_right:Ne \__enumext_renew_the_count_vii_tl
796             {
797                 \exp_not:N \renewcommand { \exp_not:V \__enumext_ref_the_count_tl } { \exp_not:V
798             }
799         }
800     }
801     \int_compare:nNt { \__enumext_keyans_level_h_int } = { 1 }
802     {
803         \tl_if_empty:NTF \__enumext_ref_key_arg_tl
804         {
805             \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
806         }
807         {
808             \tl_set_eq:NN \__enumext_ref_the_count_tl \__enumext_counter_viii_tl
809             \__enumext_regex_counter_style:
810             \tl_set_eq:NN \__enumext_ref_the_count_tl \__enumext_the_counter_viii_tl
811             \tl_put_right:Ne \__enumext_renew_the_count_viii_tl
812             {
813                 \exp_not:N \renewcommand { \exp_not:V \__enumext_ref_the_count_tl } { \exp_not:V
814             }
815         }
816     }
817 }

```

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.

```

818 \cs_new_protected:Npn \__enumext_starred_ref:
819 {
820     \int_compare:nNt { \__enumext_level_h_int } = { 1 }
821     {
822         \tl_if_empty:NF \__enumext_renew_the_count_vii_tl
823         {
824             \tl_use:N \__enumext_renew_the_count_vii_tl
825         }
826     }
827     \int_compare:nNt { \__enumext_keyans_level_h_int } = { 1 }
828     {
829         \tl_if_empty:NF \__enumext_renew_the_count_viii_tl
830         {
831             \tl_use:N \__enumext_renew_the_count_viii_tl

```

```

832     }
833   }
834 }

```

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

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

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

```

\__enumext_label_v_tl 835 \keys_define:nn { enumext / keyans }
\__enumext_label_vi_tl 836 {
837   label .code:n      = {
838                       \__enumext_label_style:cvn { \__enumext_label_v_tl }
839                       { \__enumext_counter_v_tl } {#1}
840                       \dim_set_eq:cN { \__enumext_labelwidth_v_dim }
841                       \__enumext_current_widest_dim
842                       \__enumext_label_style:cvn { \__enumext_label_vi_tl }
843                       { \__enumext_counter_vi_tl } {#1}
844                       \dim_set_eq:cN { \__enumext_labelwidth_v_dim }
845                       \__enumext_current_widest_dim
846                     },
847   label .initial:n = \Alph*,
848   label .value_required:n = true,
849   ref .code:n      = \__enumext_keyans_ref:n {#1},
850   ref .value_required:n = true,
851 }

```

(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 852 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
\__enumext_keyans_ref: 853 {
854   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
855   \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
856   {
857     \msg_error:nnn { enumext } { key-ref-empty } { keyans }
858   }
859   {
860     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
861     \__enumext_regex_counter_style:
862     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
863     \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
864     {
865       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V \l__enumext_ref_the_count_tl }
866     }
867   }
868 }

```

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

```

869 \cs_new_protected:Nn \__enumext_keyans_ref:
870 {
871   \tl_if_empty:NF \l__enumext_renew_the_count_v_tl
872   {
873     \tl_use:N \l__enumext_renew_the_count_v_tl
874   }
875 }

```

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

13.15 Setting start, start* and widest keys

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

```

\__enumext_start_from:ccn #1: \l__enumext_label_X_tl
\__enumext_start_from:cce #2: \l__enumext_start_X_int
\__enumext_start_from:cce #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.

```

876 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
877 {

```



```

878     \__enumext_if_is_int:nTF { #3 }
879     {
880         \int_set:Nn #2 {#3}
881     }
882     {
883         \regex_match:nVT { \c{Alpha} | \c{alpha} } {#1}
884         { \int_set:Nn #2 { \int_from_alph:n {#3} } }
885         \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
886         { \int_set:Nn #2 { \int_from_roman:n {#3} } }
887     }
888 }
889 \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 `\Alpha`, `\alpha`, `\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.

```

890 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
891 {
892     \__enumext_if_is_int:nTF {#4}
893     {
894         \setcounter{enumX#1} { #4 }
895     }
896     {
897         \regex_match:nVT { \c{Alpha} | \c{alpha} } {#2}
898         { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
899         \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
900         { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
901     }
902     \__enumext_label_width_by_box:cv
903     { \__enumext_labelwidth_#1_dim } { \__enumext_label_#1_tl }
904 }
905 \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.

`start`

`start*`

`widest`

```

906 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
907 {
908     \keys_define:nn { enumext / #1 }
909     {
910         start* .code:n = {
911             \__enumext_start_from:ccn
912             { \__enumext_label_#2_tl }
913             { \__enumext_start_#2_int } {##1}
914         },
915         start* .value_required:n = true,
916         start .code:n = {
917             \__enumext_start_from:cce
918             { \__enumext_label_#2_tl }
919             { \__enumext_start_#2_int } { \int_eval:n {##1} }
920         },
921         start .initial:n = 1,
922         start .value_required:n = true,
923         widest .code:n = {
924             \__enumext_widest_from:nccn {#2}
925             { \__enumext_label_#2_tl }
926             { \__enumext_labelwidth_#2_dim } {##1}
927         },
928         widest .value_required:n = true,
929     }
930 }
931 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

13.16 Setting keys for vertical spaces

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

```

topsep
partopsep
parsep
noitemsep
nosep
932 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
933 {
934   \keys_define:nn { enumext / #1 }
935   {
936     topsep .skip_set:c = { l__enumext_topsep_#2_skip },
937     topsep .initial:n = {#3},
938     topsep .value_required:n = true,
939     partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
940     partopsep .initial:n = {#4},
941     partopsep .value_required:n = true,
942     parsep .skip_set:c = { l__enumext_parsep_#2_skip },
943     parsep .initial:n = {#5},
944     parsep .value_required:n = true,
945     itemsep .skip_set:c = { l__enumext_itemsep_#2_skip },
946     itemsep .initial:n = {#6},
947     itemsep .value_required:n = true,
948     noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
949     noitemsep .value_forbidden:n = true,
950     nosep .meta:n = {
951       itemsep = 0pt, parsep = 0pt,
952       topsep = 0pt, partopsep = 0pt,
953     },
954     nosep .value_forbidden:n = true,
955   }
956 }
```

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

```

957 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
958 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
959 { 4.0pt plus 2.0pt minus 1.0pt }
960 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
961 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
962 { 2.0pt plus 1.0pt minus 1.0pt }
963 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
964 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
965 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
966 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
967 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
968 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
969 { 2.0pt plus 1.0pt minus 1.0pt }
970 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
971 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
972 { 4.0pt plus 2.0pt minus 1.0pt }
973 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
974 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
975 { 2.0pt plus 1.0pt minus 1.0pt }
```

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

13.17 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:` apply `\vspace[-\baselineskip]` and set `\topsep=0pt` for the “first level” of the nested `enumext` environment.

```

base-fix
\__enumext_nested_base_line_fix:
976 \keys_define:nn { enumext / level-1 }
977 {
978   base-fix .bool_set:N = \l__enumext_base_line_fix_bool,
979   base-fix .initial:n = false,
980   base-fix .value_forbidden:n = true,
981 }
```

The function `__enumext_nested_base_line_fix:` passed to the `__enumext_parse_keys:n` function in the definition of the `enumext` environment (§13.39) will be responsible for applying the *baseline correction* and adjusting the *⟨keys⟩* for the `enumext` environment and the `\printkeyans` with *starred argument* ‘*’ (§13.47).

We will first implement the function code from the user side of the `base-fix` key, that is, only the user knows when it is necessary to apply it within the document in which case the variable `__enumext_print_keyans_star_bool` set by the `\printkeyans` command is false and the variable `__enumext_base_line_fix_bool` is true.

```

982 \cs_new_protected:Nn \__enumext_nested_base_line_fix:
983 {
984   \bool_lazy_all:nT
985   {
986     { \bool_if_p:N \__enumext_starred_first_bool }
987     { \bool_if_p:N \__enumext_base_line_fix_bool }
988     { \bool_not_p:n { \__enumext_print_keyans_star_bool } }
989   }
990   {
991     \mode_leave_vertical:
992     \vspace { -\dim_eval:n { \baselineskip + \parsep } }
993   }

```

When we are running the `\printkeyans` command with the *starred argument* ‘*’ the variable `__enumext_print_keyans_star_bool` is true and we can run a simplified version of `\vspace` using `\skip_vertical:n`.

```

994   \bool_lazy_and:nnT
995   { \bool_if_p:N \__enumext_starred_first_bool }
996   { \bool_if_p:N \__enumext_print_keyans_star_bool }
997   {
998     \mode_leave_vertical:
999     \skip_vertical:n { -\baselineskip }
1000    \skip_vertical:N \c_zero_skip
1001  }

```

Finally we set the values of the keys `topsep`, `above` and `above*` for the “first level” of `enumext` environment equal to `0pt` and set the variable `__enumext_base_line_fix_bool` to false.

```

1002   \keys_set:nn { enumext / level-1 }
1003   {
1004     topsep = 0pt, above = 0pt, above* = 0pt,
1005   }
1006   \bool_set_false:N \__enumext_base_line_fix_bool
1007 }

```

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

13.18 Setting keys for horizontal spaces

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

```

1008 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1009 {
1010   \keys_define:nn { enumext / #1 }
1011   {
1012     itemindent .dim_set:c = { \__enumext_fake_item_indent_#2_dim },
1013     itemindent .value_required:n = true,
1014     rightmargin .dim_set:c = { \__enumext_rightmargin_#2_dim },
1015     rightmargin .value_required:n = true,
1016     listparindent .dim_set:c = { \__enumext_listparindent_#2_dim },
1017     listparindent .value_required:n = true,
1018     list-offset .dim_set:c = { \__enumext_listoffset_#2_dim },
1019     list-offset .value_required:n = true,
1020     list-indent .code:n =
1021       \bool_set_true:c { \__enumext_leftmargin_tmp_#2_bool }
1022       \dim_set:cn { \__enumext_leftmargin_tmp_#2_dim } {##1},
1023     list-indent .value_required:n = true,
1024   }
1025 }
1026 \clist_map_inline:nn
1027 {
1028   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
1029 }
1030 { \__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.

```

1031 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1032 {
1033   \keys_define:nn { enumext / #1 }
1034   {
1035     itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
1036     itemindent .value_required:n = true,
1037     rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
1038     rightmargin .value_required:n = true,
1039     listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
1040     listparindent .value_required:n = true,
1041     list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
1042     list-offset .value_required:n = true,
1043     list-indent .meta:n = { list-offset = ##1 },
1044     list-indent .value_required:n = true,
1045   }
1046 }
1047 \clist_map_inline:nn
1048 {
1049   {enumext*}{vii}, {keyans*}{viii}
1050 }
1051 { \__enumext_tmp:nn #1 }

```

13.18.1 Functions for setting the fake `itemindent`

The `itemindent` key does not set the value of `\itemindent`, it only sets the value of the *horizontal space* applied using `\skip_horizontal:N`. We will store this value in the variable and only apply it when it is greater than `0pt`. Here I will need to place `\mode_leave_vertical:` and the plain TeX macro `\ignorespaces` to avoid unwanted extra space when using the `itemindent` key.

```

1052 \cs_set_protected:Nn \__enumext_fake_item_indent:
1053 {
1054   \dim_compare:nNnT
1055   { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
1056   >
1057   { \c_zero_dim }
1058   {
1059     \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
1060     {
1061       \exp_not:N \mode_leave_vertical:
1062       \exp_not:n { \skip_horizontal:n }
1063       { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
1064       \exp_not:N \ignorespaces
1065     }
1066   }
1067 }
1068 \cs_set_protected:Nn \__enumext_keyans_fake_item_indent:
1069 {
1070   \dim_compare:nNnT
1071   { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
1072   {
1073     \tl_set:Nc \l__enumext_fake_item_indent_v_tl
1074     {
1075       \exp_not:N \mode_leave_vertical:
1076       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
1077       \exp_not:N \ignorespaces
1078     }
1079   }
1080 }
1081 \cs_set_protected:Nn \__enumext_fake_item_indent_vii:
1082 {
1083   \dim_compare:nNnT
1084   { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
1085   {
1086     \tl_set:Nc \l__enumext_fake_item_indent_vii_tl
1087     {
1088       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
1089       \exp_not:N \ignorespaces
1090     }
1091   }

```

```

1092     }
1093     \cs_set_protected:Nn \__enumext_fake_item_indent_viii:
1094     {
1095         \dim_compare:nNnT
1096         { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
1097         {
1098             \tl_set:Nc \l__enumext_fake_item_indent_viii_tl
1099             {
1100                 \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
1101                 \exp_not:N \ignorespaces
1102             }
1103         }
1104     }

```

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

13.19 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.

```

1105 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1106 {
1107     \keys_define:nn { enumext / #1 }
1108     {
1109         show-length .bool_set:c = { \l__enumext_show_length_#2_bool },
1110         show-length .initial:n = false,
1111     }
1112 }
1113 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

13.20 Setting before, after and first keys

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

```

1114 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1115 {
1116     \keys_define:nn { enumext / #1 }
1117     {
1118         before .tl_set:c = { \l__enumext_before_no_starred_key_#2_tl },
1119         before .value_required:n = true,
1120         before* .tl_set:c = { \l__enumext_before_starred_key_#2_tl },
1121         before* .value_required:n = true,
1122         after .tl_set:c = { \l__enumext_after_stop_list_#2_tl },
1123         after .value_required:n = true,
1124         first .tl_set:c = { \l__enumext_after_list_args_#2_tl },
1125         first .value_required:n = true,
1126     }
1127 }
1128 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

13.20.1 Functions for before, after and first keys in enumext

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

```

1129 \cs_new_protected:Nn \__enumext_before_args_exec:
1130 {
1131     \tl_use:c { \l__enumext_before_starred_key_ \__enumext_level: _tl }
1132 }

```

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

```

1133 \cs_new_protected:Nn \__enumext_before_keys_exec:
1134 {
1135     \tl_use:c { \l__enumext_before_no_starred_key_ \__enumext_level: _tl }
1136 }

```

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

```
1137 \cs_new_protected:Nn \__enumext_after_stop_list:
1138 {
1139     \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
1140 }
```

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

```
1141 \cs_new_protected:Nn \__enumext_after_args_exec:
1142 {
1143     \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
1144 }
```

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

13.20.2 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:
1145 \cs_new_protected:Nn \__enumext_before_args_exec_v:
1146 {
1147     \tl_use:N \l__enumext_before_starred_key_v_tl
1148 }
1149 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
1150 {
1151     \tl_use:N \l__enumext_before_no_starred_key_v_tl
1152 }
1153 \cs_new_protected:Nn \__enumext_after_stop_list_v:
1154 {
1155     \tl_use:N \l__enumext_after_stop_list_v_tl
1156 }
1157 \cs_new_protected:Nn \__enumext_after_args_exec_v:
1158 {
1159     \tl_use:N \l__enumext_after_list_args_v_tl
1160 }
```

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

13.20.3 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:
1161 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
1162 {
1163     \tl_use:N \l__enumext_before_starred_key_vii_tl
1164 }
1165 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
1166 {
1167     \tl_use:N \l__enumext_before_starred_key_viii_tl
1168 }
1169 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
1170 {
1171     \tl_use:N \l__enumext_before_no_starred_key_vii_tl
1172 }
1173 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
1174 {
1175     \tl_use:N \l__enumext_before_no_starred_key_viii_tl
1176 }
1177 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
1178 {
1179     \tl_use:N \l__enumext_after_stop_list_vii_tl
1180 }
1181 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
1182 {
1183     \tl_use:N \l__enumext_after_stop_list_viii_tl
1184 }
1185 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
1186 {
1187     \tl_use:N \l__enumext_after_list_args_vii_tl
1188 }
1189 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
1190 {
1191     \tl_use:N \l__enumext_after_list_args_viii_tl
1192 }
```


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

13.21 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.

```

1193 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1194 {
1195   \keys_define:nn { enumext / #1 }
1196   {
1197     mini-env .dim_set:c = { \__enumext_minipage_right_#2_dim },
1198     mini-env .value_required:n = true,
1199     mini-sep .dim_set:c = { \__enumext_minipage_hsep_#2_dim },
1200     mini-sep .initial:n = 0.3333em,
1201     mini-sep .value_required:n = true,
1202     columns-sep .dim_set:c = { \__enumext_columns_sep_#2_dim },
1203     columns-sep .value_required:n = true,
1204     columns .int_set:c = { \__enumext_columns_#2_int },
1205     columns .initial:n = 1,
1206     columns .value_required:n = true,
1207   }
1208 }
1209 \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.

```

1210 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1211 {
1212   \keys_define:nn { enumext / #1 }
1213   {
1214     mini-right .tl_gset:c = { g__enumext_miniright_code_#2_tl },
1215     mini-right .value_required:n = true,
1216     mini-right* .code:n = {
1217       \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
1218       \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1219     },
1220     mini-right* .value_required:n = true,
1221   }
1222 }
1223 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

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

13.22 Adjustment of vertical spaces for multicols

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

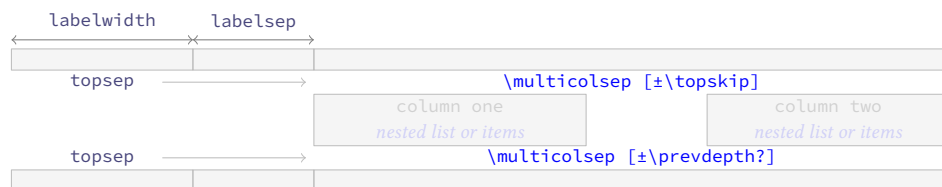


Figure 7: Representation of the vertical space in `multicols` for a nested level.

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

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

13.22.1 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 $\langle \textit{horizontal mode} \rangle$, then we will make the settings for the $\langle \textit{vertical mode} \rangle$ 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*.

```

1224 \cs_new_protected:Nn \__enumext_multi_set_vskip:
1225 {
1226   \skip_set:cn { \l__enumext_multicols_above_ \__enumext_level: _skip }
1227   {
1228     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1229   }
1230   \skip_set:cn { \l__enumext_multicols_below_ \__enumext_level: _skip }
1231   {
1232     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1233   }
1234   \__enumext_add_pre_parsep:
1235 }

```

(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*.

```

1236 \cs_new_protected:Nn \__enumext_add_pre_parsep:
1237 {
1238   \int_case:nn { \l__enumext_level_int }
1239   {
1240     { 2 }{
1241       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1242       {
1243         \skip_add:Nn \l__enumext_multicols_above_ii_skip
1244         {
1245           \l__enumext_parsep_i_skip
1246         }
1247       }
1248     }
1249     { 3 }{
1250       \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1251       {
1252         \skip_add:Nn \l__enumext_multicols_above_iii_skip
1253         {
1254           \l__enumext_parsep_ii_skip
1255         }
1256       }
1257     }
1258     { 4 }{
1259       \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1260       {
1261         \skip_add:Nn \l__enumext_multicols_above_iv_skip
1262         {
1263           \l__enumext_parsep_iii_skip
1264         }
1265       }
1266     }
1267   }
1268 }

```

(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 $\langle \textit{horizontal mode} \rangle$ or $\langle \textit{vertical mode} \rangle$.

```

1269 \cs_new_protected:Nn \__enumext_multi_addvspace:
1270 {
1271   \__enumext_multi_set_vskip:
1272   \mode_if_vertical:T
1273   {

```

```

1274     \skip_add:cn { \__enumext_multicols_above_ \__enumext_level: _skip }
1275     {
1276         \skip_use:c { \__enumext_partopsep_ \__enumext_level: _skip }
1277     }
1278     \skip_add:cn { \__enumext_multicols_below_ \__enumext_level: _skip }
1279     {
1280         \skip_use:c { \__enumext_partopsep_ \__enumext_level: _skip }
1281     }
1282 }
1283 \par\nopagebreak
1284 \addvspace{ \skip_use:c { \__enumext_multicols_above_ \__enumext_level: _skip } }
1285 }

```

(End of definition for `__enumext_multi_addvspace:`)

13.22.2 Adjustment of vertical spaces for multicols in keyans

```

\__enumext_keyans_multi_set_vskip:
\__enumext_keyans_multi_addvspace:

```

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

```

1286 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1287 {
1288     \skip_set:Nn \l__enumext_multicols_above_v_skip
1289     {
1290         \l__enumext_topsep_v_skip
1291     }
1292     \skip_set:Nn \l__enumext_multicols_below_v_skip
1293     {
1294         \l__enumext_topsep_v_skip
1295     }
1296 }
1297 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1298 {
1299     \__enumext_keyans_multi_set_vskip:
1300     \mode_if_vertical:T
1301     {
1302         \skip_add:Nn \l__enumext_multicols_above_v_skip
1303         {
1304             \skip_use:N \l__enumext_partopsep_v_skip
1305         }
1306         \skip_add:Nn \l__enumext_multicols_below_v_skip
1307         {
1308             \skip_use:N \l__enumext_partopsep_v_skip
1309         }
1310     }
1311     \par\nopagebreak
1312     \addvspace{ \l__enumext_multicols_above_v_skip }
1313 }

```

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

13.23 Adjustment of vertical spaces for minipage

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

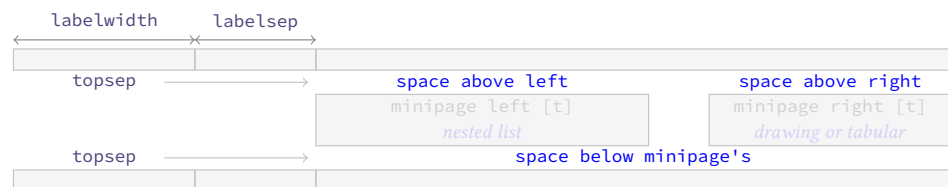


Figure 8: Representation of the `minipage` spacing adjustment for a nested level.

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

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

- Again I must make clear that the implementation here is a “*bit questionable*”, but hunting the spaces (*glue*) produced by the `minipage` environment is quite complicated, even more if `multicols` it 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.

13.23.1 Adjustment of vertical spaces for minipage in enumext

`__enumext_minipage_set_skip:`
`__enumext_minipage_add_space:`

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

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

```

1314 \cs_new_protected:Nn \__enumext_minipage_set_skip:
1315 {
1316   \skip_set:Nn \l__enumext_minipage_right_skip
1317   {
1318     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1319   }
1320   \mode_if_vertical:T
1321   {
1322     \skip_add:Nn \l__enumext_minipage_right_skip
1323     {
1324       \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1325     }
1326   }
1327   \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip

```

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

```

1328   \skip_set_eq:cN
1329   { \l__enumext_multicols_above_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1330   \skip_set_eq:cN
1331   { \l__enumext_multicols_below_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1332   \__enumext_pre_itemsep_skip:

```

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

```

1333   \int_compare:nNnT
1334   { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1335   {
1336     \skip_zero:N \topskip
1337     \skip_set_eq:Nc \multicolsep { \l__enumext_multicols_above_ \__enumext_level: _skip }
1338   }
1339 }

```

The function `__enumext_minipage_add_space:` will apply the spaces on the “*left side*” using `\addvspace` “*above*” the `__enumext_mini_page` environment, taking into account whether \TeX is in *horizontal mode* or *vertical mode*. Here we use the plain \TeX macro `\nointerlineskip` to prevent baseline “*glue*” being added between the next pair of boxes in a *vertical list*. For the latter we will make some adjustments since the `\partopsep` parameter comes into play and this affects the *vertical spacing*.

```

1340 \cs_new_protected:Nn \__enumext_minipage_add_space:
1341 {
1342   \__enumext_minipage_set_skip:
1343   \__enumext_unskip_unkern:
1344   \mode_if_vertical:TF
1345   {
1346     \nopagebreak\nointerlineskip
1347   }
1348   {
1349     \par\nopagebreak\nointerlineskip
1350     \skip_zero:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1351   }
1352   \int_compare:nNnTF
1353   { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1354   {
1355     \addvspace{ 0.445\box_ht:N \strutbox }
1356   }
1357   {
1358     \addvspace{ 0.250\box_ht:N \strutbox }
1359   }
1360 }

```

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

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

```

1361 \cs_new_protected:Nn \__enumext_pre_itemsep_skip:
1362 {
1363   \int_case:nn { \l__enumext_level_int }
1364   {
1365     { 2 }{
1366       \skip_if_eq:nnTF
1367       { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1368       {
1369         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1370         \skip_set:Nn \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1371       }
1372       {
1373         \dim_compare:nNnT
1374         { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1375         {
1376           \skip_sub:Nn
1377           \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1378           \skip_sub:Nn
1379           \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1380           \skip_add:Nn
1381           \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1382           \skip_add:Nn
1383           \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1384         }
1385         \dim_compare:nNnT
1386         { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1387         {
1388           \skip_set:Nn \l__enumext_minipage_temp_skip
1389           {
1390             \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1391           }
1392           \skip_sub:Nn
1393           \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1394           \skip_sub:Nn
1395           \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1396           \skip_add:Nn
1397           \l__enumext_minipage_after_skip
1398           { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1399           \skip_add:Nn
1400           \l__enumext_multicols_below_ii_skip
1401           { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1402         }
1403       }
1404     }
1405     { 3 }{
1406       \skip_if_eq:nnTF
1407       { \l__enumext_itemsep_ii_skip } { \c_zero_skip }
1408       {
1409         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1410         \skip_set:Nn \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1411       }
1412       {
1413         \dim_compare:nNnT
1414         { \l__enumext_itemsep_ii_skip } < { \l__enumext_minipage_after_skip }
1415         {
1416           \skip_sub:Nn
1417           \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1418           \skip_sub:Nn
1419           \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1420           \skip_add:Nn
1421           \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1422           \skip_add:Nn
1423           \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1424         }
1425         \dim_compare:nNnT
1426         { \l__enumext_itemsep_ii_skip } > { \l__enumext_minipage_after_skip }

```

```

1427         {
1428             \skip_set:Nn \l__enumext_minipage_temp_skip
1429             {
1430                 \l__enumext_itemsep_ii_skip - \l__enumext_minipage_after_skip
1431             }
1432             \skip_sub:Nn
1433             \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1434             \skip_sub:Nn
1435             \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1436             \skip_add:Nn
1437             \l__enumext_minipage_after_skip
1438             { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1439             \skip_add:Nn
1440             \l__enumext_multicols_below_iii_skip
1441             { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1442         }
1443     }
1444 }
1445 { 4 }{
1446     \skip_if_eq:nnTF { \l__enumext_itemsep_iii_skip } { \c_zero_skip }
1447     {
1448         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1449         \skip_set:Nn \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1450     }
1451     {
1452         \dim_compare:nNnT
1453         { \l__enumext_itemsep_iii_skip } < { \l__enumext_minipage_after_skip }
1454         {
1455             \skip_sub:Nn
1456             \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1457             \skip_sub:Nn
1458             \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1459             \skip_add:Nn
1460             \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1461             \skip_add:Nn
1462             \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1463         }
1464         \dim_compare:nNnT
1465         { \l__enumext_itemsep_iii_skip } > { \l__enumext_minipage_after_skip }
1466         {
1467             \skip_set:Nn \l__enumext_minipage_temp_skip
1468             {
1469                 \l__enumext_itemsep_iii_skip - \l__enumext_minipage_after_skip
1470             }
1471             \skip_sub:Nn
1472             \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1473             \skip_sub:Nn
1474             \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1475             \skip_add:Nn
1476             \l__enumext_minipage_after_skip
1477             { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1478             \skip_add:Nn
1479             \l__enumext_multicols_below_iv_skip
1480             { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1481         }
1482     }
1483 }
1484 }
1485 }

```

(End of definition for `__enumext_pre_itemsep_skip:`)

13.23.2 Adjustment of vertical spaces for minipage in keyans

```

\__enumext_keyans_minipage_set_skip:
\__enumext_keyans_minipage_add_space:
\__enumext_keyans_pre_itemsep_skip:

```

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

```

1486 \cs_new_protected:Nn \__enumext_keyans_minipage_set_skip:
1487 {
1488     \skip_zero:N \l__enumext_minipage_after_skip
1489     \skip_zero:N \l__enumext_minipage_left_skip
1490     \skip_zero:N \l__enumext_minipage_right_skip

```



```

1491 \skip_set:Nn \l__enumext_minipage_right_skip
1492 {
1493   \l__enumext_topsep_v_skip
1494 }
1495 \mode_if_vertical:T
1496 {
1497   \skip_add:Nn \l__enumext_minipage_right_skip
1498   {
1499     \l__enumext_partopsep_v_skip
1500   }
1501 }
1502 \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip
1503 \skip_set_eq:NN \l__enumext_multicols_above_v_skip \l__enumext_minipage_right_skip
1504 \skip_set_eq:NN \l__enumext_multicols_below_v_skip \l__enumext_minipage_right_skip
1505 \__enumext_keyans_pre_itemsep_skip:
1506 \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
1507 {
1508   \skip_zero:N \topskip
1509   \skip_set_eq:NN \multicolsep \l__enumext_minipage_right_skip
1510 }
1511 }
1512 \cs_new_protected:Nn \__enumext_keyans_minipage_add_space:
1513 {
1514   \__enumext_keyans_minipage_set_skip:
1515   \__enumext_unskip_unkern:
1516   \mode_if_vertical:TF
1517   {
1518     \nopagebreak\nointerlineskip
1519   }
1520   {
1521     \par\nopagebreak\nointerlineskip
1522     \skip_zero:N \l__enumext_partopsep_v_skip
1523   }
1524   \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1525   {
1526     \addvspace{ 0.445\box_ht:N \strutbox }
1527   }
1528   {
1529     \addvspace{ 0.250\box_ht:N \strutbox }
1530   }
1531 }
1532 \cs_new_protected:Nn \__enumext_keyans_pre_itemsep_skip:
1533 {
1534   \skip_if_eq:nnTF
1535   { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1536   {
1537     \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1538     \skip_set:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1539   }
1540   {
1541     \dim_compare:nNnT
1542     { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1543     {
1544       \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1545       \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1546       \skip_add:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1547       \skip_add:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1548     }
1549     \dim_compare:nNnT
1550     { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1551     {
1552       \skip_set:Nn \l__enumext_minipage_temp_skip
1553       {
1554         \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1555       }
1556       \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1557       \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1558       \skip_add:Nn \l__enumext_minipage_after_skip
1559       { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1560       \skip_add:Nn \l__enumext_multicols_below_v_skip
1561       { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }

```

```

1562     }
1563   }
1564 }

```

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

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

```

\__enumext_mini_set_vskip_vii:
\__enumext_mini_set_vskip_viii:

```

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

```

1565 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1566 {
1567   \skip_zero_new:N \l__enumext_minipage_left_skip
1568   \skip_gzero_new:N \g__enumext_minipage_right_skip
1569   \skip_gzero_new:N \g__enumext_minipage_after_skip
1570   \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1571   {
1572     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1573     \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1574   }
1575   {
1576     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1577     \skip_gset:Nn \g__enumext_minipage_right_skip
1578     {
1579       \l__enumext_topsep_vii_skip
1580     }
1581     \skip_gset:Nn \g__enumext_minipage_after_skip
1582     {
1583       0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1584     }
1585   }
1586 }
1587 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1588 {
1589   \skip_zero_new:N \l__enumext_minipage_after_skip
1590   \skip_zero_new:N \l__enumext_minipage_left_skip
1591   \skip_zero_new:N \l__enumext_minipage_right_skip
1592   \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1593   {
1594     \skip_set:Nn \l__enumext_minipage_left_skip
1595     {
1596       0.5\box_dp:N \strutbox
1597     }
1598     \skip_set:Nn \l__enumext_minipage_right_skip
1599     {
1600       \l__enumext_partopsep_viii_skip
1601     }
1602     \skip_set:Nn \l__enumext_minipage_after_skip
1603     {
1604       1.6\box_dp:N \strutbox
1605     }
1606   }
1607   {
1608     \skip_set:Nn \l__enumext_minipage_left_skip
1609     {
1610       0.5875\box_dp:N \strutbox
1611     }
1612     \skip_set:Nn \l__enumext_minipage_right_skip
1613     {
1614       \l__enumext_topsep_viii_skip
1615     }
1616     \skip_set:Nn \l__enumext_minipage_after_skip
1617     {
1618       0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1619     }
1620   }
1621 }

```

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

`__enumext_mini_addvspace_vii:`
`__enumext_mini_addvspace_viii:`

The functions `__enumext_mini_addvspace_vii:` and `__enumext_mini_addvspace_viii:` will apply the vertical space “only above” the `__enumext_mini_page` environment on the *left side* when the `mini-right` key is active in the `enumext*` and `keyans*` environments. Here we will NOT take into account whether \TeX is in *(horizontal mode)* or *(vertical mode)*, since `\partopsep` is equal to `0pt` in both environments.

```

1622 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1623 {
1624   \__enumext_mini_set_vskip_vii:
1625   \par\nopagebreak
1626   \addvspace { \l__enumext_minipage_left_skip }
1627 }
1628 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1629 {
1630   \__enumext_mini_set_vskip_viii:
1631   \par\nopagebreak
1632   \addvspace { \l__enumext_minipage_left_skip }
1633 }

```

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

13.23.4 The command `\miniright`

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

`\miniright`

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

```

1634 \NewDocumentCommand \miniright { s }
1635 {
1636   \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
1637   {
1638     \msg_error:nnn { enumext } { wrong-miniright-place }
1639   }
1640   % outside
1641   \bool_lazy_and:nnT
1642   { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
1643   { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
1644   {
1645     \msg_error:nnn { enumext } { wrong-miniright-place }
1646   }
1647   % starred env
1648   \bool_lazy_and:nnT
1649   { \bool_if_p:N \g__enumext_starred_bool }
1650   { \bool_not_p:n { \l__enumext_standar_bool } }
1651   {
1652     \msg_error:nnn { enumext } { wrong-miniright-starred }
1653   }
1654   % exec
1655   \int_compare:nNtF { \l__enumext_keyans_level_int } = { 1 }
1656   {
1657     \__enumext_keyans_mini_right_cmd:n {#1}
1658   }
1659   { \__enumext_mini_right_cmd:n {#1} }
1660 }

```

(End of definition for `\miniright`. This function is documented on page 11.)

`__enumext_mini_right_cmd:n`

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

```

1661 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1662 {
1663   \dim_compare:nNtF
1664   { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }

```

```

1665 {
1666   \__enumext_multicols_stop:
1667   \int_compare:nNnT
1668     { \int_use:c { \__enumext_columns_ \__enumext_level: _int } } = { 1 }
1669     {
1670       \par\addvspace{ \__enumext_minipage_after_skip }
1671     }
1672   \end__enumext_mini_page
1673   \hfill
1674   \__enumext_mini_page{ \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } }
1675   \par\nointerlineskip
1676   \addvspace { \__enumext_minipage_right_skip }
1677   \bool_if:nF {#1}
1678   {
1679     \centering
1680   }
1681   \int_gzero:N \g__enumext_minipage_stat_int
1682 }
1683 { \msg_error:nnn { enumext } { wrong-miniright-use } }
1684 % paranoia
1685 \RenewDocumentCommand \miniright { s }
1686 {
1687   \msg_error:nn { enumext } { many-miniright-used }
1688 }
1689 }

```

(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.

```

1690 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1691 {
1692   \dim_compare:nNnTF { \__enumext_minipage_right_v_dim } > { \c_zero_dim }
1693   {
1694     \__enumext_keyans_multicols_stop:
1695     \int_compare:nNnT { \__enumext_columns_v_int } = { 1 }
1696     {
1697       \par\addvspace{ \__enumext_minipage_after_skip }
1698     }
1699     \end__enumext_mini_page
1700     \hfill
1701     \__enumext_mini_page{ \__enumext_minipage_right_v_dim }
1702     \par\nointerlineskip
1703     \addvspace { \__enumext_minipage_right_skip }
1704     \bool_if:nF {#1}
1705     {
1706       \centering
1707     }
1708     \int_gzero:N \g__enumext_minipage_stat_int
1709   }
1710   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1711 % paranoia
1712 \RenewDocumentCommand \miniright { s }
1713 {
1714   \msg_error:nn { enumext } { many-miniright-used }
1715 }
1716 }

```

(End of definition for __enumext_keyans_mini_right_cmd:n.)

13.24 Setting above and below keys

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

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

```

1717 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2

```

below

below* ©2024 by Pablo González L

```

1718 {
1719   \keys_define:nn { enumext / #1 }
1720   {
1721     above .skip_set:c = { \__enumext_vspace_above_#2_skip },
1722     above .value_required:n = true,
1723     above* .code:n      = \bool_set_true:c { \__enumext_vspace_a_star_#2_bool }
1724                       \keys_set:nn { enumext / #1 } { above = {##1} },
1725     above* .value_required:n = true,
1726     below .skip_set:c = { \__enumext_vspace_below_#2_skip },
1727     below .value_required:n = true,
1728     below* .code:n      = \bool_set_true:c { \__enumext_vspace_b_star_#2_bool }
1729                       \keys_set:nn { enumext / #1 } { below = {##1} },
1730     below* .value_required:n = true,
1731   }
1732 }
1733 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

13.24.1 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.

```

1734 \cs_new_protected:Nn \__enumext_vspace_above:
1735 {
1736   \skip_if_eq:nnF
1737   { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1738   {
1739     \bool_if:cTF { \__enumext_vspace_a_star_ \__enumext_level: _bool }
1740     {
1741       \vspace*{ \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1742     }
1743     {
1744       \vspace { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1745     }
1746   }
1747 }

```

(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.

```

1748 \cs_new_protected:Nn \__enumext_vspace_below:
1749 {
1750   \skip_if_eq:nnF
1751   { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1752   {
1753     \bool_if:cTF { \__enumext_vspace_b_star_ \__enumext_level: _bool }
1754     {
1755       \vspace*{ \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1756     }
1757     {
1758       \vspace { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1759     }
1760   }
1761 }

```

(End of definition for `__enumext_vspace_below:`.)

13.24.2 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.

```

1762 \cs_new_protected:Nn \__enumext_vspace_above_v:
1763 {
1764   \skip_if_eq:nnF { \__enumext_vspace_above_v_skip } { \c_zero_skip }
1765   {
1766     \bool_if:NTF \__enumext_vspace_a_star_v_bool
1767     {
1768       \vspace*{ \__enumext_vspace_above_v_skip }
1769     }
1770     { \vspace { \__enumext_vspace_above_v_skip } }
1771   }
1772 }

```

(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.

```

1773 \cs_new_protected:Nn \__enumext_vspace_below_v:
1774 {
1775   \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1776   {
1777     \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1778     {
1779       \vspace*{ \l__enumext_vspace_below_v_skip }
1780     }
1781     { \vspace { \l__enumext_vspace_below_v_skip } }
1782   }
1783 }

```

(End of definition for `__enumext_vspace_below_v:`)

13.24.3 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:`

```

1784 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1785 {
1786   \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1787   {
1788     \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1789     {
1790       \vspace*{ \l__enumext_vspace_above_vii_skip }
1791     }
1792     { \vspace { \l__enumext_vspace_above_vii_skip } }
1793   }
1794 }
1795 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1796 {
1797   \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1798   {
1799     \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1800     {
1801       \vspace*{ \l__enumext_vspace_above_viii_skip }
1802     }
1803     { \vspace { \l__enumext_vspace_above_viii_skip } }
1804   }
1805 }

```

(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:`

```

1806 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1807 {
1808   \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1809   {
1810     \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1811     {
1812       \vspace*{ \l__enumext_vspace_below_vii_skip }
1813     }
1814     { \vspace { \l__enumext_vspace_below_vii_skip } }
1815   }
1816 }
1817 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1818 {
1819   \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1820   {
1821     \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1822     {
1823       \vspace*{ \l__enumext_vspace_below_viii_skip }
1824     }
1825     { \vspace { \l__enumext_vspace_below_viii_skip } }
1826   }
1827 }

```

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

13.25 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  We define the keys series, resume and resume* only for the “first level” of enumext and enumext*.
resume
resume*
1828 \cs_set_protected:Npn \__enumext_tmp:n #1
1829 {
1830   \keys_define:nn { enumext / #1 }
1831   {
1832     series .str_set:N = \__enumext_series_str,
1833     series .value_required:n = true,
1834     resume .code:n = \__enumext_resume_series:n {##1},
1835     resume* .code:n = \__enumext_resume_starred:,
1836     resume* .value_forbidden:n = true,
1837   }
1838 }
1839 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

13.25.1 Internal functions for series key

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

```

\__enumext_filter_series:n
  \__enumext_filter_series_key:n
  \__enumext_filter_series_pair:nn
1840 \cs_new:Npn \__enumext_filter_series:n #1
1841 {
1842   \use:e
1843   {
1844     \keyval_parse:NNn
1845       \__enumext_filter_series_key:n
1846       \__enumext_filter_series_pair:nn {#1}
1847   }
1848 }

```

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

```

1849 \cs_new:Npn \__enumext_filter_series_key:n #1
1850 {
1851   \str_case:nnF {#1}
1852   {
1853     { resume } {} { resume* } {} { base-fix } {}
1854   }
1855   { , { \exp_not:n {#1} } }
1856 }

```

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

```

1857 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1858 {
1859   \str_case:nnF {#1}
1860   {
1861     { series } {} { resume } {} { start } {}
1862     { start* } {} { save-ans } {} { save-key } {}
1863   }
1864   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1865 }

```

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

The function `__enumext_parse_series:n` will be responsible for storing the filtered $\langle keys \rangle$ 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 $\langle keys \rangle$. This function is passed to the function `__enumext_parse_keys:n` in the `enumext` environment definition (§13.39) and to the function `__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§13.44).

```

1866 \cs_new_protected:Npn \__enumext_parse_series:n #1
1867 {
1868   \str_if_empty:NTF \l__enumext_series_str
1869   {
1870     \bool_if:NF \l__enumext_resume_active_bool
1871     {
1872       \__enumext_resume_last:n {#1}
1873     }
1874   }
1875   {
1876     \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str _tl }
1877     \tl_gset:ce { g__enumext_series_ \l__enumext_series_str _tl }
1878     { \__enumext_filter_series:n {#1} }
1879     \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str _int }
1880     {
1881       \int_new:c { g__enumext_series_ \l__enumext_series_str _int }
1882     }
1883   }
1884 }

```

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.

```

1885 \cs_new_protected:Npn \__enumext_resume_last:n #1
1886 {
1887   \bool_if:NT \l__enumext_standar_first_bool
1888   {
1889     \tl_gclear:N \g__enumext_standar_series_tl
1890     \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1891   }
1892   \bool_if:NT \l__enumext_starred_first_bool
1893   {
1894     \tl_gclear:N \g__enumext_starred_series_tl
1895     \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1896   }
1897 }

```

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

13.25.2 Internal function to save counter value

`__enumext_resume_save_counter:`

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

- The variables `\l__enumext_series_str` and `\l__enumext__resume_name_tl` contain the same {⟨series name⟩} but are executed at different moments, the integer variable with `\l__enumext_series_str` sets the value when execute *series*={⟨series name⟩} and the integer variable with `\l__enumext__resume_name_tl` sets the subsequent values when use *resume*={⟨series name⟩}. This function is passed to the *enumext* environment definition (§13.39) and the *enumext** environment definition (§13.44).

```

1898 \cs_new_protected:Npn \__enumext_resume_save_counter:
1899 {
1900   \bool_if:NT \g__enumext_standar_bool
1901   {
1902     \tl_if_empty:NF \l__enumext_series_str
1903     {
1904       \int_gset_eq:cN
1905       { g__enumext_series_ \l__enumext_series_str _int } \value{enumXi}
1906     }
1907     \tl_if_empty:NTF \l__enumext_resume_name_tl
1908     {
1909       \str_if_empty:NF \l__enumext_series_str
1910       {
1911         \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1912       }
1913     }
1914     {
1915       \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1916       {
1917         \int_gset_eq:cN

```

```

1918         { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXi}
1919     }
1920 }
1921 \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1922 {
1923     \int_gset_eq:cN
1924     { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1925 }
1926 }
1927 \bool_if:NT \g__enumext_starred_bool
1928 {
1929     \tl_if_empty:NF \l__enumext_series_str
1930     {
1931         \int_gset_eq:cN
1932         { g__enumext_series_ \l__enumext_series_str _int } \value{enumXvii}
1933     }
1934     \tl_if_empty:NTF \l__enumext_resume_name_tl
1935     {
1936         \str_if_empty:NT \l__enumext_series_str
1937         {
1938             \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1939         }
1940     }
1941     {
1942         \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1943         {
1944             \int_gset_eq:cN
1945             { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXvii}
1946         }
1947     }
1948     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1949     {
1950         \int_gset_eq:cN
1951         { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1952     }
1953 }
1954 }

```

(End of definition for __enumext_resume_save_counter:.)

13.25.3 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.

```

1955 \cs_new_protected:Npn \__enumext_resume_series:n #1
1956 {
1957     \tl_if_empty:nTF {#1}
1958     {
1959         \__enumext_resume_counter:n { }
1960     }
1961     {
1962         \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1963         {
1964             \__enumext_resume_counter:n {#1}
1965             \bool_if:NT \g__enumext_standar_bool
1966             {
1967                 \keys_set:nv { enumext / level-1 }
1968                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1969             }
1970             \bool_if:NT \g__enumext_starred_bool
1971             {
1972                 \keys_set:nv { enumext / enumext* }
1973                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1974             }
1975         }
1976     }
1977     \bool_if:NT \g__enumext_standar_bool

```

```

1978         {
1979             \msg_error:nnn { enumext } { unknown-series } {#1}
1980         }
1981         \bool_if:NT \g__enumext_starred_bool
1982         {
1983             \msg_error:nnn { enumext } { unknown-series } {#1}
1984         }
1985     }
1986 }
1987 }

```

(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`.

```

1988 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1989 {
1990     \bool_set_true:N \l__enumext_resume_active_bool
1991     \tl_set:Nn \l__enumext_resume_name_tl {#1}
1992     \tl_if_empty:NTF \l__enumext_resume_name_tl
1993     {
1994         \__enumext_resume_counter:
1995     }
1996     {
1997         \__enumext_resume_counter_series:
1998     }
1999     \__enumext_resume_counter_save_ans:
2000 }

```

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.

```

2001 \cs_new_protected:Nn \__enumext_resume_counter:
2002 {
2003     \bool_if:NT \g__enumext_standar_bool
2004     {
2005         \int_gincr:N \g__enumext_resume_int
2006         \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
2007     }
2008     \bool_if:NT \g__enumext_starred_bool
2009     {
2010         \int_gincr:N \g__enumext_resume_vii_int
2011         \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
2012     }
2013 }

```

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.

```

2014 \cs_new_protected:Nn \__enumext_resume_counter_series:
2015 {
2016     \bool_if:NT \g__enumext_standar_bool
2017     {
2018         \int_set:Nn \l__enumext_start_i_int
2019         {
2020             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
2021         }
2022     }
2023     \bool_if:NT \g__enumext_starred_bool
2024     {
2025         \int_set:Nn \l__enumext_start_vii_int
2026         {
2027             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
2028         }
2029     }
2030 }

```

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.

```

2031 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
2032 {
2033   \bool_lazy_and:nnT
2034     { \bool_if_p:N \__enumext_standar_first_bool }
2035     { \bool_if_p:N \__enumext_store_active_bool }
2036     {
2037       \int_set:Nn \__enumext_start_i_int
2038       {
2039         \int_use:c { g__enumext_resume_ \__enumext_store_name_tl _int } + 1
2040       }
2041     }
2042   \bool_lazy_and:nnT
2043     { \bool_if_p:N \__enumext_starred_first_bool }
2044     { \bool_if_p:N \__enumext_store_active_bool }
2045     {
2046       \int_set:Nn \__enumext_start_vii_int
2047       {
2048         \int_use:c { g__enumext_resume_ \__enumext_store_name_tl _int } + 1
2049       }
2050     }
2051 }

```

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

13.25.4 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.

```

2052 \cs_new_protected:Nn \__enumext_resume_starred:
2053 {
2054   \bool_if:NT \g__enumext_standar_bool
2055   {
2056     \tl_if_empty:NF \g__enumext_standar_series_tl
2057     {
2058       \__enumext_resume_counter:n { }
2059       \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
2060     }
2061   }
2062   \bool_if:NT \g__enumext_starred_bool
2063   {
2064     \tl_if_empty:NF \g__enumext_starred_series_tl
2065     {
2066       \__enumext_resume_counter:n { }
2067       \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
2068     }
2069   }
2070 }

```

(End of definition for `__enumext_resume_starred:`.)

13.26 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.

13.26.1 Setting `save-ans` key

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

```

2071 \cs_set_protected:Npn \__enumext_tmp:n #1
2072 {
2073   \keys_define:nn { enumext / #1 }
2074   {
2075     save-ans .code:n = \__enumext_storing_set:n {##1},
2076     save-ans .value_required:n = true,
2077   }
2078 }
2079 \clist_map_inline:nn { level-1, enumext* } { { \__enumext_tmp:n {#1} } }

```

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

13.26.2 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:`.

```

2080 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
2081 {
2082   \msg_term:nnVV { enumext } { save-ans-log }
2083   \g__enumext_envir_name_tl \l__enumext_store_name_tl
2084 }
2085 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
2086 {
2087   \msg_term:nnVV { enumext } { save-ans-log-hook }
2088   \g__enumext_envir_name_tl \g__enumext_store_name_tl
2089 }

```

(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.

```

2090 \cs_new_protected:Npn \__enumext_storing_set:n #1
2091 {
2092   \tl_set:Nx \l__enumext_store_name_tl {#1}
2093   \tl_if_empty:NTF \l__enumext_store_name_tl
2094   {
2095     \bool_lazy_or:nnT
2096     { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
2097     {
2098       \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
2099     }
2100   }
2101   {
2102     \bool_lazy_or:nnT
2103     { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
2104     {
2105       \__enumext_start_save_ans_msg:
2106       \__enumext_storing_exec:
2107     }
2108   }
2109 }

```

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 `anskey*`, `keyans`, `keyans*` and `keyanspic` environments and will set to “true” the variable `\l__enumext_check_answers_bool` used for internal checking answers mechanism set by the `check-ans` and `no-store` keys, copy `{⟨store name⟩}` into the variable `\g__enumext_store_name_tl` and execute the function `__enumext_anskey_env_make:V` creating the environment `anskey*` (§13.31).

```

2110 \cs_new_protected:Nn \__enumext_storing_exec:
2111 {
2112   \bool_set_true:N \l__enumext_store_active_bool
2113   \bool_set_true:N \l__enumext_check_answers_bool
2114   \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
2115   \__enumext_anskey_env_make:V \l__enumext_store_name_tl

```

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*`.

```

2116   \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
2117   {
2118     \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
2119     \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
2120   }
2121   \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
2122   {
2123     \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
2124     \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }

```

```

2125     }
2126     \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
2127     {
2128         \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
2129         \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
2130     }
2131 }

```

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

13.26.3 The check answer mechanism

The internal mechanism for “checking answers” 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.

13.26.4 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 2132 \cs_set_protected:Npn \__enumext_tmp:n #1
no-store 2133 {
2134     \keys_define:nn { enumext / #1 }
2135     {
2136         check-ans .bool_set:N = \l__enumext_check_ans_key_bool,
2137         check-ans .initial:n = false,
2138         check-ans .value_required:n = true,
2139         no-store .code:n = {
2140             \bool_set_false:N \l__enumext_check_answers_bool
2141             \bool_set_false:N \l__enumext_check_ans_key_bool
2142         },
2143         no-store .value_forbidden:n = true,
2144     }
2145 }
2146 \clist_map_inline:nn
2147 {
2148     level-1, level-2, level-3, level-4, enumext*
2149 }
2150 { \__enumext_tmp:n {#1} }

```

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

13.26.5 Set-up check answer mechanism

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

```

2151 \cs_new_protected:Nn \__enumext_check_ans_active:
2152 {
2153   \tl_if_empty:NF \l__enumext_store_name_tl
2154   {
2155     \bool_if:NT \l__enumext_check_answers_bool
2156     {
2157       \__enumext_check_ans_level:
2158     }
2159   }
2160 }

```

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*”.

```

2161 \cs_new_protected:Nn \__enumext_check_ans_level:
2162 {
2163   \int_case:nn { \l__enumext_level_int }
2164   {
2165     { 1 }{
2166       \bool_lazy_all:nT
2167       {
2168         { \bool_if_p:N \g__enumext_starred_bool }
2169         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
2170       }
2171       {
2172         \int_gdecr:N \g__enumext_item_number_int
2173         \bool_set_false:N \l__enumext_item_number_bool
2174       }
2175     }
2176     { 2 }{
2177       \int_gdecr:N \g__enumext_item_number_int
2178       \bool_set_false:N \l__enumext_item_number_bool
2179     }
2180     { 3 }{
2181       \int_gdecr:N \g__enumext_item_number_int
2182       \bool_set_false:N \l__enumext_item_number_bool
2183     }
2184     { 4 }{
2185       \int_gdecr:N \g__enumext_item_number_int
2186       \bool_set_false:N \l__enumext_item_number_bool
2187     }
2188   }

```

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.

```

2189   \int_case:nn { \l__enumext_level_h_int }
2190   {
2191     { 1 }{
2192       \bool_lazy_all:nT
2193       {
2194         { \bool_if_p:N \g__enumext_standar_bool }
2195         { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
2196       }
2197       {
2198         \int_gdecr:N \g__enumext_item_number_int
2199         \bool_set_false:N \l__enumext_item_number_bool
2200       }
2201     }
2202   }
2203 }

```

(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.

```

2204 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
2205 {
2206   \bool_lazy_and:nnT
2207     { \bool_if_p:N \l__enumext_check_ans_key_bool }
2208     { \bool_if_p:N \g__enumext_standar_bool }
2209   {
2210     \bool_gset_true:N \g__enumext_check_ans_key_bool
2211   }
2212   \bool_lazy_and:nnT
2213     { \bool_if_p:N \l__enumext_check_ans_key_bool }
2214     { \bool_if_p:N \g__enumext_starred_bool }
2215   {
2216     \bool_gset_true:N \g__enumext_check_ans_key_bool
2217   }
2218 }

```

(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:.

```

2219 \cs_new_protected:Nn \__enumext_item_answer_diff:
2220 {
2221   \int_gset:Nn \g__enumext_item_answer_diff_int
2222     {
2223       \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
2224     }
2225 }

```

(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:.

```

2226 \cs_new_protected:Nn \__enumext_check_ans_show:
2227 {
2228   \int_case:nn { \g__enumext_item_answer_diff_int }
2229     {
2230       { -1 } { \__enumext_check_ans_msg_less: }
2231       { 0 } { \__enumext_check_ans_msg_same_ok: }
2232       { 1 } { \__enumext_check_ans_msg_greater: }
2233     }
2234 }
2235 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
2236 {
2237   \msg_warning:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2238     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2239 }
2240 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
2241 {
2242   \msg_term:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2243     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2244 }
2245 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
2246 {
2247   \msg_warning:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2248     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2249 }

```

(End of definition for __enumext_check_ans_show: and others.)

__enumext_check_ans_log:

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

__enumext_check_ans_log_msg_less:

__enumext_check_ans_log_msg_same_ok:

__enumext_check_ans_log_msg_greater:

```

2250 \cs_new_protected:Nn \__enumext_check_ans_log:
2251 {
2252   \int_case:nn { \g__enumext_item_answer_diff_int }
2253   {
2254     { -1 } { \__enumext_check_ans_log_msg_less: }
2255     { 0 } { \__enumext_check_ans_log_msg_same_ok: }
2256     { 1 } { \__enumext_check_ans_log_msg_greater: }
2257   }
2258 }
2259 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
2260 {
2261   \msg_log:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2262   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2263 }
2264 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
2265 {
2266   \msg_log:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2267   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2268 }
2269 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
2270 {
2271   \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2272   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2273 }

```

(End of definition for __enumext_check_ans_log: and others.)

13.26.6 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.

```

2274 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
2275 {
2276   \int_compare:nNnT
2277   { \g__enumext_check_starred_cmd_int } = { 0 }
2278   {
2279     \msg_warning:nnnV
2280     { enumext } { missing-starred } { #1 } \l__enumext_check_start_line_env_tl
2281   }
2282   \int_compare:nNnT
2283   { \g__enumext_check_starred_cmd_int } > { 1 }
2284   {
2285     \msg_warning:nnnV
2286     { enumext } { many-starred } { #1 } \l__enumext_check_start_line_env_tl
2287   }
2288   \int_gzero:N \g__enumext_check_starred_cmd_int
2289   \tl_clear:N \l__enumext_check_start_line_env_tl
2290 }

```

(End of definition for __enumext_check_starred_cmd:n.)

13.27 Keys and functions associated with storage

We add the keys `wrap-ans`, `wrap-opt`, `wrap-key`, `save-sep`, `mark-ans`, `mark-pos`, `mark-sep`, `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*`.

```

2291 \cs_set_protected:Npn \__enumext_tmp:n #1
2292 {
2293   \keys_define:nn { enumext / #1 }
2294   {
2295     wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
2296     wrap-ans .initial:n =
2297       {
2298         \fbox{\parbox[t]{\dimeval{\itemwidth -2\fboxsep -2\fboxrule}}{##1}}
2299       },
2300     wrap-ans .value_required:n = true,
2301     wrap-opt .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
2302     wrap-opt .initial:n = [{##1}],
2303     wrap-opt .value_required:n = true,
2304     wrap-key .cs_set_protected:Np = \__enumext_keyans_wrapper_item:n ##1,
2305     wrap-key .value_required:n = true,

```

```

2306     save-sep .tl_set:N = \__enumext_store_keyans_item_opt_sep_tl,
2307     save-sep .initial:n = {, ~ },
2308     save-sep .value_required:n = true,
2309     mark-ans .tl_set:N = \__enumext_mark_answer_sym_tl,
2310     mark-ans .initial:n = \textasteriskcentered,
2311     mark-ans .value_required:n = true,
2312     mark-pos .choice:,
2313     mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
2314     mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
2315     mark-pos / unknown .code:n =
2316         \msg_error:nnee { enumext } { unknown-choice }
2317         { mark-pos } { left, ~ right } { \exp_not:n {##1} },
2318     mark-pos .initial:n = right,
2319     mark-pos .value_required:n = true,
2320     mark-sep .dim_set:N = \__enumext_mark_sym_sep_dim,
2321     mark-sep .value_required:n = true,
2322     show-ans .bool_set:N = \__enumext_show_answer_bool,
2323     show-ans .initial:n = false,
2324     show-ans .value_required:n = true,
2325     show-pos .bool_set:N = \__enumext_show_position_bool,
2326     show-pos .initial:n = false,
2327     show-pos .value_required:n = true,
2328     mark-ref .tl_set:N = \__enumext_mark_ref_sym_tl,
2329     mark-ref .initial:n = \textreferencemark,
2330     mark-ref .value_required:n = true,
2331     save-ref .bool_set:N = \__enumext_store_ref_key_bool,
2332     save-ref .initial:n = false,
2333     save-ref .value_required:n = true,
2334 }
2335 }
2336 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {##1} }

```

(End of definition for wrap-ans and others.)

For the **keyans** and **keyans*** environments we will only add the keys mark-ans, mark-pos, mark-sep, save-sep, wrap-opt, wrap-key, show-ans and show-pos.

```

2337 \cs_set_protected:Npn \__enumext_tmp:n #1
2338 {
2339     \keys_define:nn { enumext / #1 }
2340     {
2341         mark-ans .tl_set:N = \__enumext_mark_answer_sym_tl,
2342         mark-ans .initial:n = \textasteriskcentered,
2343         mark-ans .value_required:n = true,
2344         mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
2345         mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
2346         mark-pos / unknown .code:n =
2347             \msg_error:nnee { enumext } { unknown-choice }
2348             { mark-pos } { left, ~ right } { \exp_not:n {##1} },
2349         mark-pos .initial:n = right,
2350         mark-pos .value_required:n = true,
2351         mark-pos .value_required:n = true,
2352         mark-sep .dim_set:N = \__enumext_mark_sym_sep_dim,
2353         mark-sep .value_required:n = true,
2354         save-sep .tl_set:N = \__enumext_store_keyans_item_opt_sep_tl,
2355         save-sep .value_required:n = true,
2356         wrap-opt .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
2357         wrap-opt .value_required:n = true,
2358         wrap-key .cs_set_protected:Np = \__enumext_keyans_wrapper_item:n ##1,
2359         wrap-key .value_required:n = true,
2360         show-ans .bool_set:N = \__enumext_show_answer_bool,
2361         show-ans .initial:n = false,
2362         show-ans .value_required:n = true,
2363         show-pos .bool_set:N = \__enumext_show_position_bool,
2364         show-pos .initial:n = false,
2365         show-pos .value_required:n = true,
2366     }
2367 }
2368 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {##1} }

```

(End of definition for mark-ans and others.)

13.27.1 Storing structure of the environments

The idea behind “*storing structure*” 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 argument* passed to the levels of the environment in which it is executed and “*storing*” this in the *sequence*.

```

__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 the “*storing keys*” filtered from the *optional argument* 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*.

```

2369 \cs_new_protected:Npn __enumext_store_active_keys:n #1
2370 {
2371   \bool_if:cF { __enumext_store_save_key_ __enumext_level: _bool }
2372   {
2373     \tl_clear:c { __enumext_save_key_ __enumext_level: _tl }
2374     \tl_set:ce
2375       { __enumext_store_save_key_ __enumext_level: _tl }
2376       { __enumext_filter_save_key:n {#1} }
2377   }
2378 }
2379 \cs_new_protected:Npn __enumext_store_active_keys_vii:n #1
2380 {
2381   \bool_if:NF __enumext_store_save_key_vii_bool
2382   {
2383     \tl_clear:N __enumext_store_save_key_vii_tl
2384     \tl_set:Ne __enumext_store_save_key_vii_tl { __enumext_filter_save_key:n {#1} }
2385   }
2386 }

```

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

13.27.2 Setting save-key key

Since this “*storing structure*” in the *sequence* established by the `save-ans` key when executing `__anskey` or `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 “*storing structure*” in the *sequence*.

`save-key`

The values set by this key passed in the *optional argument* 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`. Now define the key `save-key` for all levels of `enumext` and `enumext*` environments.

```

2387 \cs_set_protected:Npn __enumext_tmp:n #1
2388 {
2389   \keys_define:nn { enumext / enumext* }
2390   {
2391     save-key .code:n = __enumext_parse_save_key_vii:n {##1},
2392     save-key .value_required:n = true,
2393   }
2394   \keys_define:nn { enumext / #1 }
2395   {
2396     save-key .code:n = __enumext_parse_save_key:n {##1},
2397     save-key .value_required:n = true,
2398   }
2399 }
2400 \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 keys*” in the variable `__enumext_store_save_key_X_tl` for `enumext` and `enumext*`.

```

2401 \cs_new_protected:Npn __enumext_parse_save_key:n #1
2402 {
2403   \bool_set_true:c { __enumext_store_save_key_ __enumext_level: _bool }
2404   \tl_clear:c { __enumext_save_key_ __enumext_level: _tl }
2405   \tl_set:ce
2406     { __enumext_store_save_key_ __enumext_level: _tl }
2407     { __enumext_filter_save_key:n {#1} }
2408 }
2409 \cs_new_protected:Npn __enumext_parse_save_key_vii:n #1

```

```

2410 {
2411   \bool_set_true:N \l__enumext_store_save_key_vii_bool
2412   \tl_clear:N \l__enumext_store_save_key_vii_tl
2413   \tl_set:Nx \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2414 }

```

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

13.27.3 Internal functions to store optional arguments

The function __enumext_filter_save_key:n will be in charge of “filtering keys” we want to stored in sequence where {#1} represents the *optional argument* passed to the environment.

```

\__enumext_filter_save_key:n
  \__enumext_filter_save_key_key:n
  \__enumext_filter_save_key_pair:nn
2415 \cs_new:Npn \__enumext_filter_save_key:n #1
2416 {
2417   \use:e
2418   {
2419     \keyval_parse:NNn
2420     \__enumext_filter_save_key_key:n
2421     \__enumext_filter_save_key_pair:nn {#1}
2422   }
2423 }

```

The function __enumext_filter_save_key_key:n will be responsible for “filtering keys” that are passed “without value” by excluding the resume, resume*, no-store and base-fix keys.

```

2424 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2425 {
2426   \str_case:nnF {#1}
2427   {
2428     { resume } {} { resume* } {} { no-store } {} { base-fix } {}
2429   }
2430   { , { \exp_not:n {#1} } }
2431 }

```

The function __enumext_filter_save_key_pair:nn will be responsible for “filtering keys” that are passed “with value” by excluding the series, resume, save-ans, save-ref, save-key, check-ans, show-ans, save-pos, wrap-ans, mark-ans, wrap-opt, wrap-key, save-sep, mark-sep, mark-ref, save-sep, mini-env, mini-sep, mini-right and mini-right* keys.

```

2432 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2433 {
2434   \str_case:nnF {#1}
2435   {
2436     { series } {} { resume } {} { save-ans } {} { save-ref } {}
2437     { save-key } {} { check-ans } {} { show-ans } {} { show-pos } {}
2438     { wrap-ans } {} { mark-ans } {} { wrap-opt } {} { wrap-key } {}
2439     { save-sep } {} { mark-sep } {} { mark-ref } {} { mini-env } {}
2440     { mini-sep } {} { mini-right } {} { mini-right* } {}
2441   }
2442   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2443 }

```

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

13.27.4 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.

```

\__enumext_store_addto_prop:n
\__enumext_store_addto_prop:V
2444 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2445 {
2446   \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2447   {
2448     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2449   }
2450   { #1 }
2451 }
2452 \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V }

```

(End of definition for __enumext_store_addto_prop:n.)

13.27.5 Function for storing content in sequence

The function `__enumext_store_addto_seq:n` stores the $\{\langle content \rangle\}$ 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 $\{\langle content \rangle\}$ is stored in *sequence* is in a internal `enumext` or `enumext*` environments with the “*same structure*” in which the command was executed.

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

```

2453 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2454 {
2455     \seq_gput_right:cn { g__enumext_ \__enumext_store_name_tl _seq } { #1 }
2456 }
2457 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V }

```

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

13.27.6 Functions for storing structure in the sequence

The “*storing structure*” is handled by the functions `__enumext_store_level_open:` and `__enumext_store_level_close:` which are executed per level within the `enumext` environment.

```

2458 \cs_new_protected:Nn \__enumext_store_level_open:
2459 {
2460     \bool_if:NT \l__enumext_check_answers_bool
2461     {
2462         \tl_if_empty:cTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2463         {
2464             \__enumext_store_addto_seq:n
2465             {
2466                 \item \begin{enumext}
2467             }
2468         }
2469         {
2470             \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2471             {
2472                 \item \begin{enumext} [
2473             }
2474             \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2475             {
2476                 ]
2477             }
2478             \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2479         }
2480     }
2481 }
2482 \cs_new_protected:Nn \__enumext_store_level_close:
2483 {
2484     \bool_if:NT \l__enumext_check_answers_bool
2485     {
2486         \__enumext_store_addto_seq:n { \end{enumext} }
2487     }
2488 }

```

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

The “*storing structure*” is handled by the functions `__enumext_store_level_open_vii:` and `__enumext_store_level_close_vii:` which are executed in the `enumext*` environment.

```

2489 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2490 {
2491     \bool_if:NT \l__enumext_check_answers_bool
2492     {
2493         \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2494         {
2495             \__enumext_store_addto_seq:n
2496             {
2497                 \item \begin{enumext*}
2498             }
2499         }
2500         {
2501             \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2502             {
2503                 \item \begin{enumext*}[
2504             }

```



```

2505         \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2506         {
2507             ]
2508         }
2509         \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2510     }
2511 }
2512 }
2513 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2514 {
2515     \bool_if:NT \l__enumext_check_answers_bool
2516     {
2517         \__enumext_store_addto_seq:n { \end{enumext*} }
2518     }
2519 }

```

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

13.27.7 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

```

2520 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2521 {
2522     \mode_leave_vertical:
2523     \skip_horizontal:n { -\dim_use:N #2 }
2524     \hbox_overlap_left:n
2525     {
2526         \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2527         {
2528             \tl_use:N \l__enumext_mark_answer_sym_tl
2529         }
2530     }
2531     \skip_horizontal:n { \dim_use:N #2 }
2532 }
2533 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }

```

(End of definition for __enumext_print_keyans_box:NN.)

13.28 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.

```

2534 \cs_new_protected:Nn \__enumext_store_internal_ref:
2535 {
2536     \cs_set_protected:Npn \__enumext_tmp:n ##1
2537     {
2538         \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2539         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2540         \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2541         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2542     }
2543     \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2544     \cs_set:Npn \__enumext_tmp:n ##1
2545     { . \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.

```

2546     \bool_lazy_all:nT
2547     {
2548         { \bool_if_p:N \g__enumext_starred_bool }
2549         { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2550     }
2551     {
2552         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2553         { \tl_use:N \l__enumext_label_copy_vii_tl }

```

```

2554     }
2555     \bool_lazy_all:nT
2556     {
2557         { \bool_not_p:n { \g__enumext_standar_bool } }
2558         { \bool_if_p:N \l__enumext_standar_bool }
2559         { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } } }
2560     }
2561     {
2562         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2563         {
2564             \tl_use:N \l__enumext_label_copy_vii_tl
2565             \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2566         }
2567     }

```

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.

```

2568     \bool_lazy_all:nT
2569     {
2570         { \bool_if_p:N \g__enumext_standar_bool }
2571         { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2572         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } } }
2573     }
2574     {
2575         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2576         {
2577             \tl_use:N \l__enumext_label_copy_i_tl
2578             \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2579         }
2580     }
2581     \cs_set:Npn \__enumext_tmp:n ##1
2582     { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } . }
2583     \bool_lazy_all:nT
2584     {
2585         { \bool_if_p:N \g__enumext_standar_bool }
2586         { \bool_if_p:N \l__enumext_starred_bool }
2587         { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } } }
2588     }
2589     {
2590         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2591         {
2592             \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2593             \tl_use:N \l__enumext_label_copy_vii_tl
2594         }
2595     }

```

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

```

2596     \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2597     {
2598         \l__enumext_store_name_tl \c_colon_str
2599         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2600     }

```

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.

```

2601     \tl_put_right:Ne \l__enumext_write_aux_file_tl
2602     {
2603         \__enumext_newlabel:nn
2604         { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2605         { \l__enumext_newlabel_arg_two_tl }
2606     }
2607     \l__enumext_write_aux_file_tl
2608 }

```

(End of definition for `__enumext_store_internal_ref:.`)

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

`__enumext_store_anskey_code:n`

The internal function `__enumext_store_anskey_code:n` first we pass the $\langle \textit{argument} \rangle$ to the *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*” $\langle \textit{argument} \rangle$.

```

2609 \cs_new_protected:Npn \__enumext_store_anskey_code:n #1
2610 {
2611   \int_gincr:N \g__enumext_item_anskey_int
2612   \__enumext_store_addto_prop:n {#1}
2613   \bool_if:NT \l__enumext_store_ref_key_bool
2614   {
2615     \__enumext_store_internal_ref:
2616   }
2617   \__enumext_anskey_show_wrap_left:n { #1 }

```

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

```

2618   \tl_clear:N \l__enumext_store_anskey_arg_tl
2619   \bool_lazy_and:nnT
2620   { \bool_if_p:N \l__enumext_store_columns_break_bool }
2621   { \bool_not_p:n { \l__enumext_starred_bool } }
2622   {
2623     \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2624   }
2625   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { \item }

```

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

```

2626   \bool_lazy_and:nnT
2627   { \bool_not_p:n { \l__enumext_starred_bool } }
2628   { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2629   {
2630     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2631     {
2632       ( \exp_not:V \l__enumext_store_item_join_int )
2633     }
2634   }

```

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

```

2635   \bool_if:NTF \l__enumext_store_item_star_bool
2636   {
2637     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2638     \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2639     {
2640       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2641       {
2642         [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2643       }
2644     }
2645     \dim_compare:nT
2646     {
2647       \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2648     }
2649     {
2650       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2651       {
2652         [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2653       }
2654     }
2655     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2656   }
2657   {
2658     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2659   }

```

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

```

2660   \bool_lazy_and:nnT
2661   { \bool_if_p:N \l__enumext_store_ref_key_bool }
2662   { \bool_if_p:N \l__enumext_hyperref_bool }
2663   {
2664     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2665     {
2666       \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }

```

```

2667             { \exp_not:V \l__enumext_mark_ref_sym_tl }
2668         }
2669     }
2670     \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2671 }

```

(End of definition for __enumext_store_anskey_code:n.)

__enumext_anskey_show_wrap_arg:n

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

```

2672 \cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
2673 {
2674     \par
2675     \bool_if:NTF \l__enumext_starred_bool
2676     {
2677         \__enumext_print_keyans_box:NN
2678         \l__enumext_labelwidth_vii_dim \l__enumext_labelsep_vii_dim
2679     }
2680     {
2681         \__enumext_print_keyans_box:cc
2682         { \l__enumext_labelwidth_ \l__enumext_level: _dim }
2683         { \l__enumext_labelsep_ \l__enumext_level: _dim }
2684     }
2685     \__enumext_anskey_wrapper:n { #1 }
2686 }

```

(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 $\{\langle content \rangle\}$ stored in the prop list when using the show-pos key on the left margin next to the “wraps” $\{\langle argument \rangle\}$ passed to \anskey and the $\langle body \rangle$ in anskey* on the right side when using the show-ans key.

```

2687 \cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
2688 {
2689     \bool_if:NT \l__enumext_show_answer_bool
2690     {
2691         \__enumext_anskey_show_wrap_arg:n { #1 }
2692     }
2693     \bool_if:NT \l__enumext_show_position_bool
2694     {
2695         \tl_set:Nx \l__enumext_mark_answer_sym_tl
2696         {
2697             \group_begin:
2698             \exp_not:N \normalfont
2699             \exp_not:N \footnotesize [ \int_eval:n
2700             {
2701                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2702             }
2703             ]
2704             \group_end:
2705         }
2706         \__enumext_anskey_show_wrap_arg:n { #1 }
2707     }
2708 }

```

(End of definition for __enumext_anskey_show_wrap_left:n.)

13.30 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[$\langle key = val \rangle$]{ $\langle content \rangle$ }.

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

```

2709 \keys_define:nn { enumext / anskey }
2710 {
2711     break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2712     break-col .default:n = true,
2713     break-col .value_forbidden:n = true,
2714     item-join .int_set:N = \l__enumext_store_item_join_int,

```

```

2715     item-join .value_required:n = true,
2716     item-star .bool_set:N = \l__enumext_store_item_star_bool,
2717     item-star .default:n = true,
2718     item-star .value_forbidden:n = true,
2719     item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
2720     item-sym* .value_required:n = true,
2721     item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
2722     item-pos* .value_required:n = true,
2723     unknown .code:n = { \l__enumext_anskey_unknown:n {#1} },
2724 }

```

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

```

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

```

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

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

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

```

2739 \NewDocumentCommand \anskey { o +m }
2740 {
2741     \l__enumext_anskey_safe_outer:
2742     \group_begin:
2743         \bool_if:NT \l__enumext_check_answers_bool
2744         {
2745             \tl_if_novalue:nF {#1}
2746             {
2747                 \keys_set:nn { enumext / anskey } {#1}
2748             }
2749             \tl_if_blank:nTF {#2}
2750             {
2751                 \msg_error:nn { enumext } { anskey-empty-arg }
2752             }
2753             {
2754                 \l__enumext_anskey_safe_inner:
2755                 \l__enumext_store_anskey_code:n {#2}
2756             }
2757         }
2758     \group_end:
2759 }

```

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

13.30.1 Internal functions for the command

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

`\l__enumext_anskey_safe_inner:`

```

2760 \cs_new_protected:Nn \l__enumext_anskey_safe_outer:
2761 {
2762     \bool_if:NF \l__enumext_store_active_bool
2763     {
2764         \msg_error:nnnn { enumext } { anskey-wrong-place } { anskey } { enumext }
2765     }

```

```

2766 \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
2767 {
2768   \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
2769 }
2770 \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
2771 {
2772   \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans* }
2773 }
2774 \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
2775 {
2776   \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
2777 }
2778 }

```

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.

```

2779 \cs_new_protected:Nn \__enumext_anskey_safe_inner:
2780 {
2781   \int_incr:N \l__enumext_anskey_level_int
2782   \int_compare:nNt { \l__enumext_anskey_level_int } > { 1 }
2783   {
2784     \msg_error:nn { enumext } { anskey-nested }
2785   }
2786   \bool_if:NF \l__enumext_item_number_bool
2787   {
2788     \msg_error:nn { enumext } { anskey-unnumber-item }
2789   }
2790   \mode_if_math:T
2791   {
2792     \msg_error:nne { enumext } { anskey-math-mode } { \c_backslash_str anskey }
2793   }
2794 }

```

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

13.31 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:` (§13.32) which is executed after the environment in which the key `save-ans` is active.

```

2795 \cs_new_protected:Nn \__enumext_undefine_anskey_env:
2796 {
2797   \cs_undefine:c { anskey* }
2798   \cs_undefine:c { endanskey* }
2799   \cs_undefine:c { __scontents_anskey*_env_begin: }
2800   \cs_undefine:c { __scontents_anskey*_env_end: }
2801 }

```

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

```

2802 \__enumext_before_env:nn { enumext }
2803 {
2804   \bool_lazy_and:nnT
2805   { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2806   { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2807   {
2808     \cs_if_free:cf { __scontents_anskey*_env_begin: }
2809     {

```

```

2810         \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2811     }
2812 }
2813 }
2814 \__enumext_before_env:nn { enumext* }
2815 {
2816     \bool_lazy_and:nnT
2817     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2818     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2819     {
2820         \cs_if_free:cF { __scontents_anskey*_env_begin: }
2821         {
2822             \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2823         }
2824     }
2825 }

```

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.

```

2826 \__enumext_before_env:nn { anskey* }
2827 {
2828     \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2829     {
2830         \msg_error:nnn { enumext } { anskey-env-wrong } { keyans }
2831     }
2832     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2833     {
2834         \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
2835     }
2836     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2837     {
2838         \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
2839     }
2840     \bool_if:NF \l__enumext_item_number_bool
2841     {
2842         \msg_error:nn { enumext } { anskey-unnumber-item }
2843     }
2844     \mode_if_math:T
2845     {
2846         \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
2847     }
2848 }

```

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

```

2849 \cs_new_protected:Npn \__enumext_anskey_env_make:n #1
2850 {
2851     \bool_if:NT \l__enumext_anskey_env_bool
2852     {
2853         \newenvsc{anskey*}[store-env=#1,print-env=false]
2854         \__enumext_anskey_env_exec:
2855     }
2856 }
2857 \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`.

```

2858 \cs_new_protected:Nn \__enumext_anskey_env_define_keys:
2859 {
2860     \keys_define:nn { scontents / scontents }
2861     {
2862         break-col .bool_gset:N = \g__enumext_store_columns_break_bool,
2863         break-col .default:n = true,

```



```

2864         break-col .value_forbidden:n = true,
2865         item-join .int_gset:N = \g__enumext_store_item_join_int,
2866         item-join .value_required:n = true,
2867         item-star .bool_gset:N = \g__enumext_store_item_star_bool,
2868         item-star .default:n = true,
2869         item-star .value_forbidden:n = true,
2870         item-sym* .tl_gset:N = \g__enumext_store_item_symbol_tl,
2871         item-sym* .value_required:n = true,
2872         item-pos* .dim_gset:N = \g__enumext_store_item_symbol_sep_dim,
2873         item-pos* .value_required:n = true,
2874         print-env .undefine:,
2875         store-env .undefine:,
2876         write-out .undefine:,
2877         unknown .code:n = { \__enumext_anskey_env_unknown:n {##1} },
2878     }
2879 }

```

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`.

```

2880 \cs_new_protected:Npn \__enumext_anskey_env_unknown:n #1
2881 {
2882     \exp_args:NV \__enumext_anskey_env_unknown:nn \l_keys_key_str {#1}
2883 }
2884 \cs_new_protected:Npn \__enumext_anskey_env_unknown:nn #1#2
2885 {
2886     \tl_if_blank:nTF {#2}
2887     {
2888         \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
2889     }
2890     {
2891         \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
2892     }
2893 }

```

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`.

```

2894 \cs_new_protected:Nn \__enumext_anskey_env_reset_keys:
2895 {
2896     \keys_define:nn { scontents / scontents }
2897     {
2898         break-col .undefine:,
2899         item-join .undefine:,
2900         item-star .undefine:,
2901         item-sym* .undefine:,
2902         item-pos* .undefine:,
2903         write-out .code:n = {
2904             \bool_set_false:N \l__scontents_storing_bool
2905             \bool_set_true:N \l__scontents_writing_bool
2906             \tl_set:Nn \l__scontents_fname_out_tl {##1}
2907         },
2908         write-out .value_required:n = true,
2909         print-env .meta:nn = { scontents } { print-env = ##1 },
2910         print-env .default:n = true,
2911         store-env .meta:nn = { scontents } { store-env = ##1 },
2912         unknown .code:n = { \__scontents_parse_environment_keys:n {##1} },
2913     }
2914 }

```

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*.

```

2915 \cs_new_protected:Npn \__enumext_rescan_anskey_env:n #1
2916 {
2917     \group_begin:
2918     \int_set:Nn \tex_newlinechar:D { `^^J }
2919     \__scontents_rescan_tokens:x
2920     {
2921         \endgroup % This assumes \catcode`\=0... Things might go off otherwise.
2922         #1
2923     }
2924 }

```

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

`__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)`.

```
2925 \cs_new_protected:Nn \__enumext_anskey_env_exec:
2926 {
2927   \__enumext_before_env:nn { anskey* }
2928   {
2929     \__enumext_anskey_env_define_keys:
2930   }
```

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.

```
2931   \hook_if_empty:nF {env/anskey*/after}
2932   {
2933     \hook_gremove_code:nn {env/anskey*/after} { * }
2934   }
2935   \__enumext_after_env:nn { anskey* }
2936   {
2937     \__enumext_anskey_env_save_keys:
2938     \tl_clear:N \l__enumext_store_anskey_env_tl
2939     \tl_clear:N \l__enumext_store_anskey_opt_tl
2940     \bool_if:NT \l__enumext_check_answers_bool
2941     {
2942       \tl_gset:Ne \l__enumext_store_anskey_env_tl
2943       {
2944         \seq_item:ce { g__scontents_name_ \l__enumext_store_name_tl _seq } { -1 }
2945       }
2946       \regex_match:nVTF
2947       { ^\s* \z | ^\s* \u{c__scontents_hidden_space_str} \z }
2948       \l__enumext_store_anskey_env_tl
2949       {
2950         \msg_error:nn { enumext } { anskey-empty-arg }
2951       }
2952       {
2953         \__enumext_anskey_env_store:
2954       }
2955     }
2956     \__enumext_anskey_env_clean_vars:
2957     \__enumext_anskey_env_reset_keys:
2958   }
2959 }
```

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`.

```
2960 \cs_new_protected:Nn \__enumext_anskey_env_save_keys:
2961 {
2962   \bool_lazy_and:nnT
2963   { \bool_if_p:N \g__enumext_store_columns_break_bool }
2964   { \bool_not_p:n { \l__enumext_starred_bool } }
2965   {
2966     \tl_put_left:Ne \l__enumext_store_anskey_opt_tl { ,break-col, }
2967   }
```

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

```
2968   \bool_lazy_and:nnT
2969   { \bool_not_p:n { \l__enumext_starred_bool } }
2970   { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }
2971   {
2972     \tl_put_left::Ne \l__enumext_store_anskey_opt_tl
2973     {
2974       ,item-join = \exp_not:V \g__enumext_store_item_join_int,
2975     }
2976   }
```

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

```

2977   \bool_if:NT \g__enumext_store_item_star_bool
2978   {
2979     \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2980     {
2981       ,item-star,
2982     }
2983     \tl_if_empty:NF \g__enumext_store_item_symbol_tl
2984     {
2985       \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2986       {
2987         ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,
2988       }
2989     }
2990     \dim_compare:nT
2991     {
2992       \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
2993     }
2994     {
2995       \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2996       {
2997         ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
2998       }
2999     }
3000   }
3001 }

```

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`.

```

3002 \cs_new_protected:Nn \__enumext_anskey_env_store:
3003 {
3004   \group_begin:
3005   \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
3006   {
3007     \exp_args:Ne
3008     \__enumext_store_anskey_code:n
3009     {
3010       \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
3011     }
3012   }
3013   {
3014     \keys_set_known:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
3015     \exp_args:Ne
3016     \__enumext_store_anskey_code:n
3017     {
3018       \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
3019     }
3020   }
3021   \group_end:
3022 }

```

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

```

3023 \cs_new_protected:Nn \__enumext_anskey_env_clean_vars:
3024 {
3025   \bool_gset_false:N \g__enumext_store_columns_break_bool
3026   \int_gzero:N       \g__enumext_store_item_join_int
3027   \bool_gset_false:N \g__enumext_store_item_star_bool
3028   \tl_gclear:N       \g__enumext_store_item_symbol_tl
3029   \dim_gzero:N       \g__enumext_store_item_symbol_sep_dim
3030 }

```

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

13.32 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*` (§13.31) 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.

```

3031 \cs_new_protected:Nn \__enumext_execute_after_env:
3032 {
3033   \int_compare:nNt { \__enumext_level_int } = { 0 }
3034   {
3035     \tl_if_empty:NF \g__enumext_store_name_tl
3036     {
3037       \__enumext_stop_save_ans_msg:
3038       \__enumext_item_answer_diff:
3039       \__enumext_log_global_vars:
3040       \__enumext_log_answer_vars:
3041       \bool_if:NTF \g__enumext_check_ans_key_bool
3042       {
3043         \__enumext_check_ans_show:
3044       }
3045       { \__enumext_check_ans_log: }
3046       \__enumext_undefine_anskey_env:
3047     }
3048     \__enumext_reset_global_vars:
3049   }
3050 }

```

(End of definition for `__enumext_execute_after_env:`.)

- This function is passed to the function `__enumext_after_env:nn` for the environments `enumext` (§13.39) and `enumext*` (§13.44) and it is executed only when the environments are not nested or at some level of these..

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

13.33.1 Storing content in prop list

`__enumext_keyans_addto_prop:n`

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

```

3051 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
3052 {
3053   \tl_clear:N \__enumext_store_current_label_tl
3054   \int_compare:nNtTF { \__enumext_keyans_pic_level_int } = { 1 }
3055   {
3056     \tl_put_right:Ne \__enumext_store_current_label_tl { \__enumext_label_vi_tl }
3057   }
3058   {
3059     \tl_put_right:Ne \__enumext_store_current_label_tl { \__enumext_label_v_tl }
3060   }

```

If the *optional argument* is present and the `save-sep` key is not empty, we save it.

```

3061   \tl_if_novalue:nF { #1 }
3062   {
3063     \tl_if_empty:NF \__enumext_store_keyans_item_opt_sep_tl
3064     {
3065       \tl_put_right:Ne \__enumext_store_current_label_tl
3066       {
3067         \__enumext_store_keyans_item_opt_sep_tl
3068       }
3069     }
3070     \tl_put_right:Ne \__enumext_store_current_label_tl { #1 }
3071   }
3072   \__enumext_store_addto_prop:V \__enumext_store_current_label_tl
3073 }

```

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

13.33.2 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 `\anskey` basically because in this environments the interest is in the current `<label>` for `\item*` and `\anspic*` with the `<contents>` of the *optional argument*. The mechanism defined here will allow to execute `\ref{<store name : position>}` and will return `1 . (A)`.

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

```

3074 \cs_new_protected:Nn \__enumext_keyans_store_ref:
3075 {
3076   \bool_if:NT \__enumext_store_ref_key_bool
3077   {
3078     \cs_set_protected:Npn \__enumext_tmp:n ##1
3079     {
3080       \tl_set_eq:cc { \__enumext_label_copy_##1_tl } { \__enumext_label_##1_tl }
3081       \tl_reverse:c { \__enumext_label_copy_##1_tl }
3082       \tl_remove_once:cn { \__enumext_label_copy_##1_tl } { . }
3083       \tl_reverse:c { \__enumext_label_copy_##1_tl }
3084     }
3085     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
3086     \__enumext_keyans_store_ref_aux_i:
3087   }
3088 }

```

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

```

3089 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
3090 {
3091   \bool_if:NT \g__enumext_starred_bool
3092   {
3093     \tl_set_eq:NN \__enumext_label_copy_i_tl \__enumext_label_copy_vii_tl
3094   }
3095   \int_compare:nNt { \__enumext_keyans_pic_level_int } = { 1 }
3096   {
3097     \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
3098     { \__enumext_label_copy_i_tl . \__enumext_label_copy_vi_tl }
3099   }
3100   \int_compare:nNt { \__enumext_keyans_level_int } = { 1 }
3101   {
3102     \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
3103     { \__enumext_label_copy_i_tl . \__enumext_label_copy_v_tl }
3104   }
3105   \int_compare:nNt { \__enumext_keyans_level_h_int } = { 1 }
3106   {
3107     \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
3108     { \__enumext_label_copy_i_tl . \__enumext_label_copy_viii_tl }
3109   }
3110   \tl_put_right:Ne \__enumext_newlabel_arg_one_tl
3111   {
3112     \__enumext_store_name_tl \c_colon_str
3113     \int_eval:n { \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop } }
3114   }
3115   \__enumext_keyans_store_ref_aux_ii:
3116 }

```

Now auxiliary function `__enumext_keyans_store_ref_aux_ii:` save the result in the variable `__enumext_write_aux_file_tl` and finally we write in the .aux file.

```

3117 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
3118 {
3119   \tl_put_right:Ne \__enumext_write_aux_file_tl
3120   {
3121     \__enumext_newlabel:nn
3122     { \exp_not:V \__enumext_newlabel_arg_one_tl }
3123     { \__enumext_newlabel_arg_two_tl }
3124   }
3125   \__enumext_write_aux_file_tl
3126 }

```

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

13.33.3 Storing content in sequence

`__enumext_keyans_addto_seq:n` The function `__enumext_keyans_addto_seq:n` will pass the contents of the current `\label` `__enumext_label_v_tl` for the `keyans` environment and the `__enumext_label_vi_tl` for the `keyanspic` environment when using `\item*` and `\anspic*`, followed by the `\contents` of the *optional argument* of both

commands to the `\l__enumext_store_current_label_tl` variable to the sequence defined by the `save-ans` key.

```

3127 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
3128 {
3129   \tl_clear:N \l__enumext_store_current_label_tl
3130   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
3131   {
3132     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
3133   }
3134   {
3135     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
3136   }
3137   \tl_if_novalue:nF { #1 }
3138   {
3139     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
3140     {
3141       \tl_put_right:Ne \l__enumext_store_current_label_tl
3142       {
3143         \l__enumext_store_keyans_item_opt_sep_tl
3144       }
3145     }
3146     \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
3147   }
3148   \__enumext_keyans_addto_seq_link:
3149 }

```

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.

```

3150 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
3151 {
3152   \bool_lazy_and:nnT
3153   { \bool_if_p:N \l__enumext_store_ref_key_bool }
3154   { \bool_if_p:N \l__enumext_hyperref_bool }
3155   {
3156     \tl_put_right:Ne \l__enumext_store_current_label_tl
3157     {
3158       \hfill \exp_not:N \hyperlink
3159       {
3160         \exp_not:V \l__enumext_newlabel_arg_one_tl
3161       }
3162       { \exp_not:V \l__enumext_mark_ref_sym_tl }
3163     }
3164   }
3165   \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
3166   \bool_if:NT \l__enumext_check_answers_bool
3167   {
3168     \int_gincr:N \g__enumext_item_anskey_int
3169   }
3170 }

```

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

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

`__enumext_keyans_save_item_opt:n`
`__enumext_keyans_show_item_opt:`

The function `__enumext_keyans_save_item_opt:n` will save the optional argument of `\item*` and `\anspic*` in the variable `\l__enumext_store_current_opt_arg_tl`.

```

3171 \cs_new_protected:Npn \__enumext_keyans_save_item_opt:n #1
3172 {
3173   \tl_if_novalue:nF { #1 }
3174   {
3175     \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
3176   }
3177 }

```

The function `__enumext_keyans_show_item_opt:` will print the optional arguments of `\item*` and `\anspic*` when the `show-ans` or `show-pos` keys are set.

```

3178 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
3179 {

```

```

3180     \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
3181     {
3182         \bool_lazy_or:nnT
3183         { \bool_if_p:N \l__enumext_show_answer_bool }
3184         { \bool_if_p:N \l__enumext_show_position_bool }
3185         {
3186             \__enumext_keyans_wrapper_opt:n
3187             { \l__enumext_store_current_opt_arg_tl } \c_space_tl
3188         }
3189     }
3190 }

```

(End of definition for `__enumext_keyans_save_item_opt:n` and `__enumext_keyans_show_item_opt:.`)

```

\__enumext_keyans_pos_mark_set:
\__enumext_keyans_show_ans:
\__enumext_keyans_show_pos:

```

The function `__enumext_keyans_pos_mark_set:` adjusts the horizontal spaces for the `mark-sep` key taking into account the value of the `align` key and the width of `\label`.

```

3191 \cs_new_protected:Nn \__enumext_keyans_pos_mark_set:
3192 {
3193     \__enumext_label_width_by_box:Nn
3194     \l__enumext_mark_sep_tmpa_dim { \l__enumext_label_v_tl }
3195     \str_case:Vn \l__enumext_align_label_pos_v_str
3196     {
3197         { l }
3198         {
3199             \dim_set:Nn \l__enumext_mark_sep_tmpb_dim { \c_zero_dim }
3200         }
3201         { r }
3202         {
3203             \dim_set:Nn \l__enumext_mark_sep_tmpb_dim
3204             { \l__enumext_labelwidth_v_dim - \l__enumext_mark_sep_tmpa_dim }
3205         }
3206         { c }
3207         {
3208             \dim_set:Nn \l__enumext_mark_sep_tmpb_dim
3209             { 0.5\l__enumext_labelwidth_v_dim - 0.5\l__enumext_mark_sep_tmpa_dim }
3210         }
3211     }

```

Here we set the default value for the key `mark-sep`.

```

3212     \dim_compare:nNnT { \l__enumext_mark_sym_sep_dim } = { \c_zero_dim }
3213     {
3214         \dim_set:Nn \l__enumext_mark_sym_sep_dim { \l__enumext_labelsep_v_dim }
3215     }
3216     \dim_add:Nn \l__enumext_mark_sym_sep_dim { \l__enumext_mark_sep_tmpb_dim }
3217 }

```

The function `__enumext_keyans_show_ans:` will print the `\symbol` set by the `mark-ans` key when the `show-ans` key is active.

```

3218 \cs_new_protected:Nn \__enumext_keyans_show_ans:
3219 {
3220     \bool_lazy_all:nT
3221     {
3222         { \bool_if_p:N \l__enumext_show_answer_bool }
3223         { \bool_if_p:N \l__enumext_item_wrap_key_bool }
3224     }
3225     {
3226         \__enumext_keyans_pos_mark_set:
3227         \__enumext_print_keyans_box:NN
3228         \l__enumext_labelwidth_v_dim \l__enumext_mark_sym_sep_dim
3229     }
3230 }

```

The function `__enumext_keyans_show_pos:` will print the `\position` of the stored content in *prop list*. Need add `1` to `\g__enumext_<store name>_prop` for `keyans` environment.

```

3231 \cs_new_protected:Nn \__enumext_keyans_show_pos:
3232 {
3233     \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
3234     {
3235         \int_incr:N \l__enumext_show_pos_tmp_int
3236     }
3237     {
3238         \int_zero:N \l__enumext_show_pos_tmp_int

```



```

3239     }
3240     \bool_lazy_all:nT
3241     {
3242         { \bool_if_p:N \l__enumext_show_position_bool }
3243         { \bool_if_p:N \l__enumext_item_wrap_key_bool }
3244     }
3245     {
3246         \tl_set:Nx \l__enumext_mark_answer_sym_tl
3247         {
3248             \group_begin:
3249             \exp_not:N \normalfont
3250             \exp_not:N \footnotesize [ \int_eval:n
3251             {
3252                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + \l__enumext_sho
3253             }
3254             ]
3255             \group_end:
3256         }
3257         \__enumext_keyans_pos_mark_set:
3258         \__enumext_print_keyans_box:NN
3259         \l__enumext_labelwidth_v_dim \l__enumext_mark_sym_sep_dim
3260     }
3261 }

```

(End of definition for `__enumext_keyans_pos_mark_set:`, `__enumext_keyans_show_ans:`, and `__enumext_keyans_show_pos:`.)

13.34 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.

When *labeling* PDF is active `\makelabel` is redefined as `\hss #1` and the only way to get the `align` key to work correctly is to redefine `\makelabel` using `\makebox`. The best way to implement this is to use the conditional command `\IfDocumentMetadataTF` to force this redefinition and the dedicated `mode-box` key to manually activate it by the user.

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`.

```

3262 \cs_new_protected:Npn \__enumext_default_item:n #1
3263 {
3264     \tl_if_novalue:nTF {#1}
3265     {
3266         \bool_if:NT \l__enumext_check_answers_bool
3267         {
3268             \int_gincr:N \g__enumext_item_number_int
3269             \bool_set_true:N \l__enumext_item_number_bool
3270         }
3271         \bool_set_true:c { \l__enumext_wrap_label_ \__enumext_level: _bool }
3272         \__enumext_item_std:w \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
3273     }
3274     {
3275         \bool_set_eq:cc
3276         { \l__enumext_wrap_label_ \__enumext_level: _bool }
3277         { \l__enumext_wrap_label_opt_ \__enumext_level: _bool }
3278         \__enumext_item_std:w [#1] \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
3279     }
3280 }

```

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

`__enumext_starred_item_exec:nn`

`__enumext_starred_item_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`.

```
3281 \cs_new_protected:Npn \__enumext_starred_item_exec:nn #1 #2
3282 {
3283   \tl_if_novalue:nTF {#1}
3284   {
3285     \tl_gset_eq:Nc
3286     \g__enumext_item_symbol_aux_tl { \l__enumext_item_symbol_ \__enumext_level: _tl }
3287   }
3288   {
3289     \tl_gset:Nn \g__enumext_item_symbol_aux_tl {#1}
3290   }
3291   \tl_if_novalue:nTF {#2}
3292   {
3293     \dim_set_eq:cc
3294     { \l__enumext_item_symbol_sep_ \__enumext_level: _dim }
3295     { \l__enumext_labelsep_ \__enumext_level: _dim }
3296   }
3297   {
3298     \dim_set:cn { \l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
3299   }
3300   \bool_if:NT \l__enumext_check_answers_bool
3301   {
3302     \int_gincr:N \g__enumext_item_number_int
3303     \bool_set_true:N \l__enumext_item_number_bool
3304   }
3305   \bool_set_true:c { \l__enumext_wrap_label_ \__enumext_level: _bool }
3306   \__enumext_item_std:w \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
3307 }
```

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

```
3308 \cs_new_protected:Nn \__enumext_starred_item_exec:
3309 {
3310   \tl_if_empty:cF { \l__enumext_item_symbol_ \__enumext_level: _tl }
3311   {
3312     \mode_leave_vertical:
3313     \skip_horizontal:n { -\dim_use:c { \l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3314     \hbox_overlap_left:n { \g__enumext_item_symbol_aux_tl }
3315     \skip_horizontal:n { \dim_use:c { \l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3316   }
3317 }
```

(End of definition for `__enumext_starred_item_exec:nn` and `__enumext_starred_item_exec:.`)

`__enumext_redefine_item:`

The function `__enumext_redefine_item:` will redefine the `\item` command in the `enumext` environment adding `\item*`. This function are passed to `__enumext_list_arg_two_X:` used in the definition of the `enumext` environment (§13.39).

```
3318 \cs_new_protected:Nn \__enumext_redefine_item:
3319 {
3320   \RenewDocumentCommand \item { s o o }
3321   {
3322     \bool_if:nTF {##1}
3323     {
3324       \__enumext_starred_item_exec:nn {##2} {##3}
3325     }
3326     { \__enumext_default_item:n {##2} }
3327   }
3328 }
```

(End of definition for `__enumext_redefine_item:.`)

`__enumext_make_label:` The function `__enumext_make_label:` redefine `\makelabel` for the keys `mode-box`, `align`, `font`, `wrap-label`, `wrap-label*` and `\item*` for `enumext` environment. This function are passed to `__enumext_list_arg_two_X:` used in the definition of the `enumext` environment (§13.39).

```

3329 \cs_new_protected:Nn \__enumext_make_label:
3330 {
3331   \IfDocumentMetadataTF
3332   {
3333     \__enumext_make_label_box:
3334   }
3335   {
3336     \bool_if:NTF \l__enumext_mode_box_bool
3337     {
3338       \__enumext_make_label_box:
3339     }
3340     {
3341       \__enumext_make_label_std:
3342     }
3343   }
3344 }

```

Standard definition when `\DocumentMetadata` is not active.

```

3345 \cs_new_protected:Nn \__enumext_make_label_std:
3346 {
3347   \RenewDocumentCommand \makelabel { m }
3348   {
3349     \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
3350     \__enumext_starred_item_exec:
3351     \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3352     \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3353     {
3354       \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3355     }
3356     { ##1 }
3357     \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
3358     \tl_gclear:N \g__enumext_item_symbol_aux_tl
3359   }
3360 }

```

Definition using `\makebox` when `\DocumentMetadata` is active or `mode-box` is active.

- Here it is necessary to use `\strut\smash` to maintain text *alignment* in case the user wants to use `\labelbx` for example. In my experiments with *mimicking* the `description` environment it was the only way out and it seems to have no adverse effects and may serve in the future as a basis for a more generic `list` environment package than `enumext`.

```

3361 \cs_new_protected:Nn \__enumext_make_label_box:
3362 {
3363   \RenewDocumentCommand \makelabel { m }
3364   {
3365     \strut\smash
3366     {
3367       \makebox
3368       [ \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim } ]
3369       [ \str_use:c { l__enumext_align_label_pos_ \__enumext_level: _str } ]
3370       {
3371         \__enumext_starred_item_exec:
3372         \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3373         \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3374         {
3375           \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3376         }
3377         { ##1 }
3378         \tl_gclear:N \g__enumext_item_symbol_aux_tl
3379       }
3380     } % close smash
3381   }
3382 }

```

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

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

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

```

item-sym* Define and set item-sym* and item-pos* keys for enumext and enumext*.
item-pos*
3383 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
3384 {
3385   \keys_define:nn { enumext / #1 }
3386   {
3387     item-sym* .tl_set:c = { \__enumext_item_symbol_#2_tl },
3388     item-sym* .value_required:n = true,
3389     item-sym* .initial:n = {\textborn},
3390     item-pos* .dim_set:c = { \__enumext_item_symbol_sep_#2_dim },
3391     item-pos* .value_required:n = true,
3392   }
3393 }
3394 \clist_map_inline:nn
3395 {
3396   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
3397 }
3398 { \__enumext_tmp:nn #1 }

```

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

13.36 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).

13.36.1 Handling unknown keys for keyans, keyans* and keyanspic

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

```

\__enumext_keyans_unknown_keys:n
\__enumext_keyans_unknown_keys:nn
3399 \cs_set_protected:Npn \__enumext_tmp:n #1
3400 {
3401   \keys_define:nn { enumext / #1 }
3402   {
3403     unknown .code:n = { \__enumext_keyans_unknown_keys:n {##1} }
3404   }
3405 }
3406 \clist_map_inline:nn { keyans, keyans*, keyanspic } { \__enumext_tmp:n {#1} }

```

Internal functions for handling unknown key.

```

3407 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:n #1
3408 {
3409   \exp_args:NV \__enumext_keyans_unknown_keys:nn \l_keys_key_str {#1}
3410 }
3411 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:nn #1#2
3412 {
3413   \tl_if_blank:nTF {#2}
3414   {
3415     \msg_error:nne { enumext } { keyans-unknown-key } {#1}
3416   }
3417   {
3418     \msg_error:nnee { enumext } { keyans-unknown-key-value } {#1} {#2}
3419   }
3420 }

```

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

13.36.2 Handling unknown keys for enumext*

Define and set unknown key for enumext* environment.

```

\__enumext_starred_unknown_keys:n
\__enumext_starred_unknown_keys:nn
3421 \keys_define:nn { enumext / enumext* }
3422 {
3423   unknown .code:n = { \__enumext_starred_unknown_keys:n {#1} }
3424 }

```

Internal functions for handling unknown key.

```

3425 \cs_new_protected:Npn \__enumext_starred_unknown_keys:n #1
3426 {
3427   \exp_args:NV \__enumext_starred_unknown_keys:nn \l_keys_key_str {#1}
3428 }
3429 \cs_new_protected:Npn \__enumext_starred_unknown_keys:nn #1#2
3430 {
3431   \tl_if_blank:nTF {#2}
3432   {
3433     \msg_error:nnn { enumext } { starred-unknown-key } {#1}

```

```

3434     }
3435     {
3436         \msg_error:nnnn { enumext } { starred-unknown-key-value } {#1} {#2}
3437     }
3438 }

```

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

13.36.3 Handling unknown keys for enumext

`unknown` Defines and set the key `unknown` for `enumext` environment.

```

\__enumext_standar_unknown_keys:n
\__enumext_standar_unknown_keys:nn
3439 \cs_set_protected:Npn \__enumext_tmp:n #1
3440 {
3441     \keys_define:nn { enumext / #1 }
3442     {
3443         unknown .code:n = { \__enumext_standar_unknown_keys:n {##1} }
3444     }
3445 }
3446 \list_map_inline:nn { level-1,level-2,level-3,level-4 } { \__enumext_tmp:n {#1} }

```

Internal functions for handling `unknown` key.

```

3447 \cs_new_protected:Npn \__enumext_standar_unknown_keys:n #1
3448 {
3449     \exp_args:NV \__enumext_standar_unknown_keys:nn \l_keys_key_str {#1}
3450 }
3451 \cs_new_protected:Npn \__enumext_standar_unknown_keys:nn #1#2
3452 {
3453     \tl_if_blank:nTF {#2}
3454     {
3455         \msg_error:nnn { enumext } { standar-unknown-key } {#1}
3456     }
3457     {
3458         \msg_error:nnnn { enumext } { standar-unknown-key-value } {#1} {#2}
3459     }
3460 }

```

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

13.37 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` along with the keys `wrap-label`, `wrap-label*` and `itemindent`.

```

3461 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
3462 {
3463     \tl_if_novalue:nTF { #1 }
3464     {
3465         \bool_set_true:N \__enumext_wrap_label_v_bool
3466         \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl
3467     }
3468     {
3469         \bool_set_eq:NN \__enumext_wrap_label_v_bool \__enumext_wrap_label_opt_v_bool
3470         \__enumext_item_std:w [#1] \tl_use:N \__enumext_fake_item_indent_v_tl
3471     }
3472 }

```

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

`__enumext_keyans_starred_item:n` The function `__enumext_keyans_starred_item:n` will take as argument *#1* the *optional argument* [*⟨content⟩*] passed to `\item*` and save it via the `__enumext_keyans_save_item_opt:n` function, then activate the `wrap-label` key, execute `\item` using `__enumext_item_std:w`, the `itemindent` key and print the *optional argument* using the `__enumext_keyans_show_item_opt:` function handled by the `wrap-opt` key.

```

3473 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
3474 {
3475     \__enumext_keyans_save_item_opt:n { #1 }
3476     \bool_set_true:N \__enumext_wrap_label_v_bool
3477     \__enumext_item_std:w \tl_use:N \__enumext_fake_item_indent_v_tl
3478     \__enumext_keyans_show_item_opt:

```

Now *store* the current *(label)* first in the *prop list* (including the *optional argument*), run the internal “*label and ref*” system if the *save-ref* key is active, then *store* in the *sequence* and finally increments `\g__enumext_-check_starred_cmd_int` for internal check system.

```

3479     \__enumext_keyans_addto_prop:n { #1 }
3480     \__enumext_keyans_store_ref:
3481     \__enumext_keyans_addto_seq:n { #1 }
3482     \int_gincr:N \g__enumext_check_starred_cmd_int
3483 }

```

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

```

\item*
\__enumext_keyans_redefine_item:

```

The function `__enumext_keyans_redefine_item:` is responsible for adding the *starred argument* and *optional argument* by the `__enumext_list_arg_two_v:` function in the definition of the *keyans* environment. Here we will set to true the variable `\l__enumext_item_wrap_key_bool` used by the *wrap-key* key only when `\item*` is executed and additionally we need to use `\peek_remove_spaces:n` to avoid an unwanted space when using `\item*` together with the *itemindent* key. This function are passed to `__enumext_list_arg_two_v:` used in the definition of the *keyans* environment (§13.38).

```

3484 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
3485 {
3486   \RenewDocumentCommand \item { s o }
3487   {
3488     \bool_if:nTF {##1}
3489     {
3490       \bool_set_true:N \l__enumext_item_wrap_key_bool % wrap-key
3491       \peek_remove_spaces:n
3492       {
3493         \__enumext_keyans_starred_item:n {##2}
3494       }
3495     }
3496     {
3497       \bool_set_false:N \l__enumext_item_wrap_key_bool
3498       \__enumext_keyans_default_item:n {##2}
3499     }
3500   }
3501 }

```

(End of definition for `\item*` and `__enumext_keyans_redefine_item:`. This function is documented on page 15.)

```

\__enumext_keyans_make_label:
\__enumext_keyans_wrapper_label:n
\__enumext_keyans_make_label_std:
\__enumext_keyans_make_label_box:

```

The function `__enumext_keyans_make_label:` redefine `\makeLabel` for the keys *mode-box*, *align*, *font*, *wrap-label*, *wrap-label**, *wrap-key* and `\item*` for *keyans* environment. This function are passed to `__enumext_list_arg_two_v:` used in the definition of the *keyans* environment (§13.38).

```

3502 \cs_new_protected:Nn \__enumext_keyans_make_label:
3503 {
3504   \IfDocumentMetadataTF
3505   {
3506     \__enumext_keyans_make_label_box:
3507   }
3508   {
3509     \bool_if:NTF \l__enumext_mode_box_bool
3510     {
3511       \__enumext_keyans_make_label_box:
3512     }
3513     {
3514       \__enumext_keyans_make_label_std:
3515     }
3516   }
3517 }

```

We added conditionals to the `__enumext_keyans_wrapper_label:n` function to handle the keys *wrap-key*, *wrap-label* and *wrap-label**.

```

3518 \cs_new_protected:Npn \__enumext_keyans_wrapper_label:n #1
3519 {
3520   \bool_lazy_all:nT
3521   {
3522     { \bool_if_p:N \l__enumext_wrap_label_v_bool }
3523     { \bool_if_p:N \l__enumext_show_answer_bool }
3524     { \bool_if_p:N \l__enumext_item_wrap_key_bool }
3525     { \cs_if_exist_p:N \__enumext_keyans_wrapper_item:n }
3526   }
3527 {

```

```

3528     \cs_set_eq:NN \__enumext_wrapper_label_v:n \__enumext_keyans_wrapper_item:n
3529   }
3530   \bool_if:NTF \l__enumext_wrap_label_v_bool
3531   {
3532     \__enumext_wrapper_label_v:n { #1 }
3533   }
3534   { #1 }
3535 }

```

Standard definition when `\DocumentMetadata` is not active.

```

3536 \cs_new_protected:Nn \__enumext_keyans_make_label_std:
3537 {
3538   \RenewDocumentCommand \makeLabel { m }
3539   {
3540     \tl_use:N \l__enumext_label_fill_left_v_tl
3541     \__enumext_keyans_show_ans:
3542     \__enumext_keyans_show_pos:
3543     \tl_use:N \l__enumext_label_font_style_v_tl
3544     \__enumext_keyans_wrapper_label:n { ##1 }
3545     \tl_use:N \l__enumext_label_fill_right_v_tl
3546   }
3547 }

```

Definition using `\makebox` when `\DocumentMetadata` is active or `mode-box` is active.

```

3548 \cs_new_protected:Nn \__enumext_keyans_make_label_box:
3549 {
3550   \RenewDocumentCommand \makeLabel { m }
3551   {
3552     \strut\smash
3553     {
3554       \makebox[ \l__enumext_labelwidth_v_dim ][ \l__enumext_align_label_pos_v_str ]
3555       {
3556         \__enumext_keyans_show_ans:
3557         \__enumext_keyans_show_pos:
3558         \tl_use:N \l__enumext_label_font_style_v_tl
3559         \__enumext_keyans_wrapper_label:n { ##1 }
3560       }
3561     }
3562   }
3563 }

```

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

13.38 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`.

13.38.1 Calculation of `\leftmargin` and `\itemindent`

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

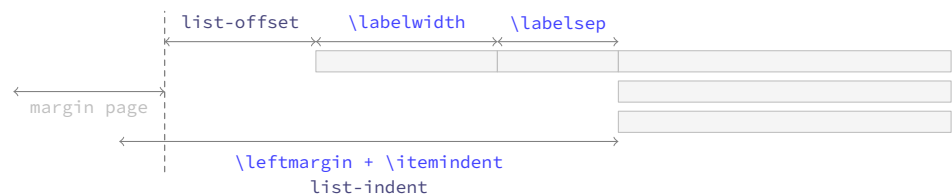


Figure 9: Representation of standard horizontal lengths in `list` environment.

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

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

```

\__enumext_calc_hspace:NNNNNNN
\__enumext_calc_hspace:ccccc

```

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

```

#1: \l__enumext_labelwidth_X_dim      #2: \l__enumext_labelsep_X_dim
#3: \l__enumext_listoffset_X_dim     #4: \l__enumext_leftmargin_tmp_X_dim

```

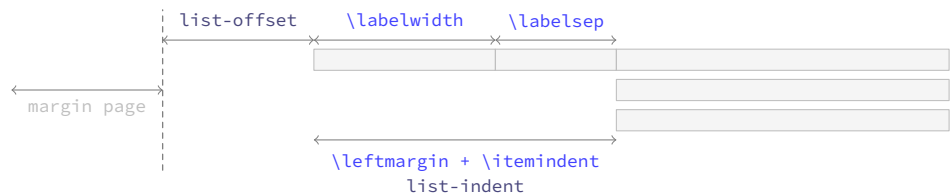



Figure 10: Representation of horizontal lengths concept in list in enumext.

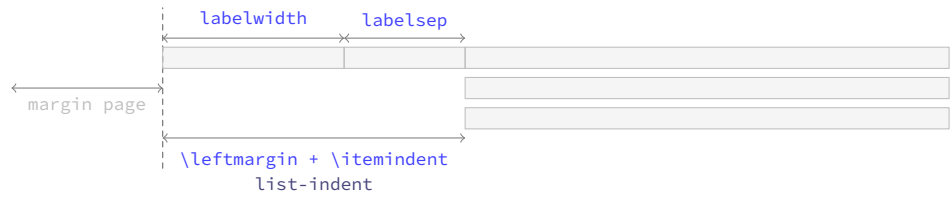


Figure 11: Default horizontal lengths in enumext.

```
#5: \l__enumext_leftmargin_X_dim      #6: \l__enumext_itemindent_X_dim
#7: \l__enumext_leftmargin_tmp_X_bool
```

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

This function is passed to `__enumext_list_arg_two_X`: which is used in the definition of the `enumext` and `keyans` environments (§13.38).

```
3564 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNN #1 #2 #3 #4 #5 #6 #7
3565 {
3566   \dim_compare:nNnT { #1 } < { \c_zero_dim }
3567   {
3568     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
3569     \dim_set:Nn #1 { \dim_abs:n { #1 } }
3570   }
3571   \dim_compare:nNnT { #2 } < { \c_zero_dim }
3572   {
3573     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
3574     \dim_set:Nn #2 { \dim_abs:n { #2 } }
3575   }
3576 }
```

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

```
3576 \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }
```

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

```
3577 \dim_compare:nNnTF { #4 } < { \c_zero_dim }
3578 {
3579   \dim_set:Nn #6 { #1 + #2 - #4 }
3580   \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3581 }
3582 {
3583   \dim_compare:nNnT { #4 } = { #1 + #2 }
3584   { \dim_set:Nn #6 { \c_zero_dim } }
3585   \dim_compare:nNnT { #4 } < { #1 + #2 }
3586   { \dim_set:Nn #6 { #1 + #2 - #4 } }
3587   \dim_compare:nNnT { #4 } > { #1 + #2 }
3588   {
3589     \dim_set:Nn #6 { -#1 - #2 + #4 }
3590     \dim_set:Nn #6 { #6*-1 }
3591   }
3592   \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3593 }
3594 }
3595 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { ccccccc }
```

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

13.38.2 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.

```
3596 \cs_set_protected:Npn \__enumext_tmp:n #1
3597 {
3598   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
```

```

3599 {
3600   \__enumext_calc_hspace:ccccc
3601   { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3602   { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3603   { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3604   { \__enumext_leftmargin_tmp_#1_bool }
3605   \clist_map_inline:nn
3606     { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3607     { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
3608   \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3609     { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
3610   \usecounter { enumX#1 }
3611   \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
3612   \str_if_eq:nnTF {#1} { v }
3613   {
3614     \__enumext_keyans_redefine_item:
3615     \__enumext_keyans_make_label:
3616     \__enumext_keyans_ref:
3617     \__enumext_keyans_fake_item_indent:
3618     \bool_if:cT { \__enumext_show_length_#1_bool }
3619     {
3620       \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3621     }
3622   }
3623   {
3624     \__enumext_redefine_item:
3625     \__enumext_make_label:
3626     \__enumext_standar_ref:
3627     \__enumext_fake_item_indent:
3628     \bool_if:cT { \__enumext_show_length_#1_bool }
3629     {
3630       \msg_term:nnne { enumext } { list-lengths } {#1}
3631       { \int_use:N \__enumext_level_int }
3632     }
3633   }
3634 }
3635 }
3636 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

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

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

```

3637 \cs_set_protected:Npn \__enumext_tmp:n #1
3638 {
3639   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3640   {
3641     \bool_set_true:c { \__enumext_leftmargin_tmp_#1_bool }
3642     \dim_zero:c { \__enumext_leftmargin_tmp_#1_dim }
3643     \__enumext_calc_hspace:ccccc
3644     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3645     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3646     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3647     { \__enumext_leftmargin_tmp_#1_bool }
3648     \clist_map_inline:nn
3649       { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3650       { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
3651     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3652       { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
3653     \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
3654     \skip_zero:N \partopsep
3655     \usecounter { enumX#1 }
3656     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
3657     \__enumext_starred_ref:
3658     \str_if_eq:nnTF {#1} { vii }
3659     {
3660       \__enumext_fake_item_indent_vii:
3661       \bool_if:cT { \__enumext_show_length_vii_bool }
3662       { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }

```

```

3663     }
3664     {
3665         \__enumext_fake_item_indent_viii:
3666         \bool_if:cT { \__enumext_show_length_#1_bool }
3667             { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3668     }
3669 }
3670 }
3671 \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:.)

13.39 The environment enumext

__enumext_safe_exec: The __enumext_safe_exec: function first call the function __enumext_is_not_nested: which sets \g__enumext_standar_bool to “true” if we are NOT nested within `enumext*`, then call the function __enumext_internal_mini_page: to create the environment `__enumext_mini_page`, we will increment \l__enumext_level_int to restrict nesting of the environment, set \l__enumext_standar_bool to “true” and finally call the function __enumext_is_on_first_level: which sets \l__enumext_standar_first_bool to “true” only if the environment is NOT nested and we are at the “first level”.

```

3672 \cs_new_protected:Nn \__enumext_safe_exec:
3673 {
3674     \__enumext_is_not_nested:
3675     \__enumext_internal_mini_page:
3676     \int_incr:N \l__enumext_level_int
3677     \int_compare:nNnT { \l__enumext_level_int } > { 4 }
3678         { \msg_fatal:nn { enumext } { list-too-deep } }
3679     \bool_set_true:N \l__enumext_standar_bool
3680     \bool_set_false:N \l__enumext_starred_bool
3681     \__enumext_is_on_first_level:
3682 }

```

(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 *sequence* if the key `save-key` is not active.

```

3683 \cs_new_protected:Npn \__enumext_parse_keys:n #1
3684 {
3685     \tl_if_novalue:nF {#1}
3686     {
3687         \str_clear:N \l__enumext_series_str
3688         \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
3689             {
3690                 \keys_set:nn { enumext / level-1 } {#1}
3691                 \__enumext_parse_series:n {#1}
3692                 \__enumext_nested_base_line_fix:
3693             }
3694             {
3695                 \exp_args:Ne \keys_set:nn
3696                     { enumext / level-\int_use:N \l__enumext_level_int } {#1}
3697             }
3698         \__enumext_store_active_keys:n {#1}
3699     }
3700 }

```

(End of definition for __enumext_parse_keys:n.)

__enumext_start_store_level: The __enumext_start_store_level: function activate the “storing structure” mechanism in the *sequence* for the command `\anskey` and the environment `anskey*`.

```

3701 \cs_new_protected:Nn \__enumext_start_store_level:
3702 {
3703     \bool_lazy_all:nT
3704     {
3705         { \bool_if_p:N \l__enumext_store_active_bool }
3706         { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3707         { \bool_if_p:N \g__enumext_standar_bool }
3708     }

```

```

3709     {
3710         \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3711         {
3712             \bool_set_true:c { \l__enumext_store_upper_level_ \__enumext_level: _bool }
3713             \__enumext_store_level_open:
3714         }
3715     }

```

If `enumext` are nested in `enumext*` add `__enumext_store_level_open:` to preserve the “*storing structure*”.

```

3716     \bool_lazy_all:nT
3717     {
3718         { \bool_if_p:N \l__enumext_store_active_bool }
3719         { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3720         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
3721     }
3722     {
3723         \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3724         {
3725             \bool_set_true:c { \l__enumext_store_upper_level_ \__enumext_level: _bool }
3726             \__enumext_store_level_open:
3727         }
3728     }
3729 }

```

(End of definition for `__enumext_start_store_level:`)

`__enumext_stop_store_level:` The `__enumext_stop_store_level:` function stop the “*storing structure*” mechanism in the *sequence* for the command `\anskey` and the environment `anskey*`.

```

3730 \cs_new_protected:Nn \__enumext_stop_store_level:
3731 {
3732     \bool_if:cT { \l__enumext_store_upper_level_ \__enumext_level: _bool }
3733     {
3734         \__enumext_store_level_close:
3735     }
3736 }

```

(End of definition for `__enumext_stop_store_level:`)

`__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.

```

3737 \cs_new_protected:Nn \__enumext_multicols_start:
3738 {
3739     \int_compare:nNnT
3740     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3741     {
3742         \dim_compare:nNnT
3743         { \dim_use:c { \l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3744         {
3745             \dim_set:cn { \l__enumext_columns_sep_ \__enumext_level: _dim }
3746             {
3747                 ( \dim_use:c { \l__enumext_labelwidth_ \__enumext_level: _dim }
3748                 + \dim_use:c { \l__enumext_labelsep_ \__enumext_level: _dim }
3749                 ) / \int_use:c { \l__enumext_columns_ \__enumext_level: _int }
3750                 - \dim_use:c { \l__enumext_listoffset_ \__enumext_level: _dim }
3751             }
3752         }
3753         \dim_set_eq:Nc \columnsep { \l__enumext_columns_sep_ \__enumext_level: _dim }
3754         \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3755         {
3756             \dim_zero:N \columnseprule
3757         }
3758     }

```

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.

```

3758     \bool_if:cF { \l__enumext_minipage_active_ \__enumext_level: _bool }
3759     {
3760         \skip_zero:N \multicolsep
3761         \__enumext_multi_addvspace:
3762     }
3763     \raggedcolumns
3764     \begin{multicols}{ \int_use:c { \l__enumext_columns_ \__enumext_level: _int } }

```

```

3765     }
3766 }

```

(End of definition for `__enumext_multicols_start:`)

`__enumext_multicols_stop:` The function `__enumext_multicols_stop:` will stop the `multicols` environment and apply our “vertical adjust” spacing. For compatibility with *tagged* PDF, the closing of the `list` environment is executed here along with `__enumext_stop_store_level:`.

```

3767 \cs_new_protected:Nn \__enumext_multicols_stop:
3768 {
3769   \int_compare:nNnTF
3770   { \int_use:c { \__enumext_columns_ \__enumext_level: _int } } > { 1 }
3771   {
3772     \__enumext_stop_list:
3773     \__enumext_stop_store_level:
3774     \end{multicols}
3775     \__enumext_unskip_unkern:
3776     \__enumext_unskip_unkern:
3777     \par\addvspace{ \skip_use:c { \__enumext_multicols_below_ \__enumext_level: _skip } }
3778   }
3779   {
3780     \__enumext_stop_list:
3781     \__enumext_stop_store_level:
3782   }
3783 }

```

(End of definition for `__enumext_multicols_stop:`)

`__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.

```

3784 \cs_new_protected:Nn \__enumext_before_list:
3785 {
3786   \__enumext_vspace_above:
3787   \__enumext_before_args_exec:
3788   \__enumext_check_ans_active:

```

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

```

3789   \dim_compare:nNnT
3790   { \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3791   {
3792     \dim_set:cn { \__enumext_minipage_left_ \__enumext_level: _dim }
3793     {
3794       \linewidth
3795       - \dim_use:c { \__enumext_minipage_right_ \__enumext_level: _dim }
3796       - \dim_use:c { \__enumext_minipage_hsep_ \__enumext_level: _dim }
3797     }

```

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

```

3798     \bool_set_true:c { \__enumext_minipage_active_ \__enumext_level: _bool }
3799     \int_gincr:N \g__enumext_minipage_stat_int
3800     \__enumext_minipage_add_space:
3801     \noindent
3802     \__enumext_mini_page{ \dim_use:c { \__enumext_minipage_left_ \__enumext_level: _dim } }
3803   }
3804   \__enumext_multicols_start:
3805 }

```

(End of definition for `__enumext_before_list:`)

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

```

3806 \cs_new_protected:Nn \__enumext_second_part:
3807 {
3808   \bool_if:cTF { \l__enumext_minipage_active_ \__enumext_level: _bool }
3809   {
3810     \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
3811     {
3812       \msg_warning:nn { enumext } { missing-miniright }
3813       \miniright
3814     }
3815     \int_gzero:N \g__enumext_minipage_stat_int
3816     \__enumext_unskip_unkern: % remove topsep + [partopsep]
3817     \end__enumext_mini_page
3818   }
3819   {
3820     \__enumext_multicols_stop:
3821   }

```

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.

```

3822   \__enumext_after_stop_list:
3823   \__enumext_check_ans_key_hook:
3824   \__enumext_vspace_below:
3825   \bool_set_false:N \l__enumext_standar_bool
3826   \__enumext_resume_save_counter:
3827 }

```

(End of definition for `__enumext_second_part:`.)

`__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.

```

3828 \cs_new_protected:Nn \__enumext_set_item_width:
3829 {
3830   \dim_set:Nn \itemwidth { \linewidth }
3831   \dim_compare:nT
3832   {
3833     \dim_use:c { \l__enumext_listoffset_ \__enumext_level: _dim } != \c_zero_dim
3834   }
3835   {
3836     \dim_sub:Nn \itemwidth
3837     {
3838       \dim_use:c { \l__enumext_listoffset_ \__enumext_level: _dim }
3839     }
3840   }
3841 }

```

(End of definition for `__enumext_set_item_width:`.)

enumext Now create the `enumext` environment based on `list` environment by levels.

```

3842 \NewDocumentEnvironment{enumext}{0}{}
3843 {
3844   \__enumext_safe_exec:
3845   \__enumext_parse_keys:n {#1}
3846   \__enumext_before_list:
3847   \__enumext_start_store_level:
3848   \__enumext_start_list:nn
3849   { \tl_use:c { \l__enumext_label_ \__enumext_level: _tl } }
3850   {
3851     \use:c { __enumext_list_arg_two_ \__enumext_level: : }
3852     \__enumext_before_keys_exec:
3853   }
3854   \__enumext_set_item_width:
3855   \__enumext_after_args_exec:
3856 }
3857 {

```

```

3858     \__enumext_second_part:
3859 }

```

(End of definition for *enumext*. This function is documented on page 5.)

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`.

```

3860 \__enumext_after_env:nn {enumext}
3861 {
3862     \__enumext_execute_after_env:
3863 }

```

13.40 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”.

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.

```

3864 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3865 {
3866     \bool_if:NF \l__enumext_store_active_bool
3867     {
3868         \msg_error:nnnn { enumext } { wrong-place } { keyans } { save-ans }
3869     }
3870     \int_incr:N \l__enumext_keyans_level_int
3871     \bool_set_true:N \l__enumext_keyans_env_bool
3872     \__enumext_keyans_name_and_start:
3873     % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3874     \bool_set_false:N \l__enumext_store_active_bool
3875     \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
3876     {
3877         \msg_error:nn { enumext } { keyans-nested }
3878     }
3879     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3880     {
3881         \msg_error:nn { enumext } { keyans-wrong-level }
3882     }
3883 }

```

(End of definition for `__enumext_keyans_safe_exec:.`)

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

```

3884 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3885 {
3886     \keys_set:nn { enumext / keyans } {#1}
3887 }

```

(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_second_part_v:
3888 \cs_new_protected:Nn \__enumext_before_list_v:
3889 {
3890     \__enumext_vspace_above_v:
3891     \__enumext_before_args_exec_v:
3892     \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3893     {
3894         \dim_set:Nn \l__enumext_minipage_left_v_dim
3895         {
3896             \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3897         }
3898         \bool_set_true:N \l__enumext_minipage_active_v_bool
3899         \int_gincr:N \g__enumext_minipage_stat_int
3900         \__enumext_keyans_minipage_add_space:
3901         \__enumext_mini_page{ \l__enumext_minipage_left_v_dim }
3902     }
3903     \__enumext_keyans_multicols_start:
3904 }
3905 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3906 {

```



```

3907 \int_compare:nNt { \l__enumext_columns_v_int } > { 1 }
3908 {
3909   \dim_compare:nNt { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3910   {
3911     \dim_set:Nn \l__enumext_columns_sep_v_dim
3912     {
3913       (
3914         \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3915       ) / \l__enumext_columns_v_int
3916       - \l__enumext_listoffset_v_dim
3917     }
3918   }
3919   \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim
3920   \dim_zero:N \columnseprule % no rule here
3921   \bool_if:NF \l__enumext_minipage_active_v_bool
3922   {
3923     \skip_zero:N \multicolsep
3924     \__enumext_keyans_multi_addvspace:
3925   }
3926   \raggedcolumns
3927   \begin{multicols}{ \l__enumext_columns_v_int }
3928 }
3929 }
3930 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3931 {
3932   \int_compare:nNt { \l__enumext_columns_v_int } > { 1 }
3933   {
3934     \__enumext_stop_list:
3935     \end{multicols}
3936     \__enumext_unskip_unkern:
3937     \__enumext_unskip_unkern:
3938     \par\addvspace{ \l__enumext_multicols_below_v_skip }
3939   }
3940   {
3941     \__enumext_stop_list:
3942   }
3943 }
3944 \cs_new_protected:Nn \__enumext_second_part_v:
3945 {
3946   \bool_if:NTF \l__enumext_minipage_active_v_bool
3947   {
3948     \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
3949     {
3950       \msg_warning:nn { enumext } { missing-miniright }
3951       \miniright
3952     }
3953     \int_gzero:N \g__enumext_minipage_stat_int
3954     \__enumext_unskip_unkern: % remove \topsep + [\partopsep]
3955     \end__enumext_mini_page
3956     \par\addvspace{ \l__enumext_minipage_after_skip }
3957   }
3958   {
3959     \__enumext_keyans_multicols_stop:
3960   }
3961   \bool_set_false:N \l__enumext_keyans_env_bool
3962   \__enumext_after_stop_list_v:
3963   \__enumext_vspace_below_v:
3964 }

```

(End of definition for __enumext_before_list_v: and others.)

__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.

```

3965 \cs_new_protected:Nn \__enumext_keyans_set_item_width:
3966 {
3967   \dim_set:Nn \itemwidth { \linewidth }
3968   \dim_compare:nT
3969   {
3970     \l__enumext_listoffset_v_dim != \c_zero_dim
3971   }
3972   {

```

```

3973         \dim_sub:Nn \itemwidth { \l__enumext_listoffset_v_dim }
3974     }
3975 }

```

(End of definition for `__enumext_keyans_set_item_width:`)

keyans Now we define the environment **keyans** also based on lists.

```

3976 \NewDocumentEnvironment{keyans}{0}{ }
3977 {
3978     \__enumext_keyans_safe_exec:
3979     \__enumext_keyans_parse_keys:n {#1}
3980     \__enumext_before_list_v:
3981     \__enumext_start_list:nn
3982     { \tl_use:N \l__enumext_label_v_tl }
3983     {
3984         \__enumext_list_arg_two_v:
3985         \__enumext_before_keys_exec_v:
3986     }
3987     \__enumext_keyans_set_item_width:
3988     \__enumext_after_args_exec_v:
3989 }
3990 {
3991     \__enumext_check_starred_cmd:n { item }
3992     \__enumext_second_part_v:
3993 }

```

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

13.41 Tagging PDF support for non-standart list environments

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

To maintain a correct `list` structure when `\DocumentMetadata` is active, it is necessary to do some things manually using `tagpdf`[17] and `ltsockets`[19]. This implementation is an adaptation of my answer thanks to Ulrike Fischer's comments in [How can I modify my \item redefinition to be compatible with tagging-pdf](#).

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

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

```

start-list-tags
stop-start-tags
stop-list-tags
__enumext_start_list_tag:n
    \__enumext_stop_start_list_tag:
__enumext_stop_list_tag:n
3994 \socket_new:nn {tagsupport/__enumext/starred}{1 }
3995 \socket_new_plugin:nnn {tagsupport/__enumext/starred} {start-list-tags}
3996 {
3997     \tag_resume:n {#1}
3998     \tag_mc_end_push:
3999     \tag_struct_begin:n {tag=LI}
4000     \tag_struct_begin:n {tag=Lbl}
4001     \tag_mc_begin:n {tag=Lbl}
4002 }
4003 \socket_new_plugin:nnn {tagsupport/__enumext/starred} {stop-start-tags}
4004 {
4005     \tag_mc_end:
4006     \tag_struct_end:n {tag=Lbl}
4007     \tag_struct_begin:n {tag=LBody}
4008     \tag_struct_begin:n {tag=text-unit}
4009     \tag_struct_begin:n {tag=text}
4010 }
4011 \socket_new_plugin:nnn {tagsupport/__enumext/starred} {stop-list-tags}
4012 {
4013     \tag_struct_end:n {tag=text}
4014     \tag_struct_end:n {tag=text-unit}
4015     \tag_struct_end:n {tag=LBody}
4016     \tag_struct_end:n {tag=LI}
4017     \tag_mc_begin_pop:n {}
4018     \tag_suspend:n {#1}
4019 }

```

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

```

4020 \cs_new_protected_nopar:Npn \__enumext_start_list_tag:n #1
4021 {

```

```

4022     \IfDocumentMetadataTF
4023     {
4024         \socket_assign_plug:nn {tagsupport/__enumext/starred} {start-list-tags}
4025         \socket_use:nn {tagsupport/__enumext/starred} {#1}
4026     } {}
4027 }
4028 \cs_new_protected_nopar:Nn \__enumext_stop_start_list_tag:
4029 {
4030     \IfDocumentMetadataTF
4031     {
4032         \socket_assign_plug:nn {tagsupport/__enumext/starred} {stop-start-tags}
4033         \socket_use:nn {tagsupport/__enumext/starred} { }
4034     } {}
4035 }
4036 \cs_new_protected_nopar:Npn \__enumext_stop_list_tag:n #1
4037 {
4038     \IfDocumentMetadataTF
4039     {
4040         \socket_assign_plug:nn {tagsupport/__enumext/starred} {stop-list-tags}
4041         \socket_use:nn {tagsupport/__enumext/starred} {#1}
4042     } {}
4043 }

```

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

13.41.2 Socket for tagging support in keyanspic

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

```

start-list-tags
stop-start-tags
stop-list-tags
__enumext_anspic_start_list_tag:
__enumext_anspic_stop_start_list_tag:
__enumext_anspic_stop_list_tag:
4044 \socket_new:nn {tagsupport/__enumext/keyanspic}{ 0 }
4045 \socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {start-list-tags}
4046 {
4047     \tag_resume:n {keyanspic}
4048     \tag_mc_end_push:
4049         \tag_struct_begin:n {tag=LI}
4050         \tag_struct_begin:n {tag=Lbl}
4051         \tag_mc_begin:n {tag=Lbl}
4052 }
4053 \socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {stop-start-tags}
4054 {
4055     \tag_mc_end:
4056     \tag_struct_end:n {tag=Lbl}
4057     \tag_struct_begin:n {tag=LBody}
4058     \tag_struct_begin:n {tag=text-unit}
4059     \tag_struct_begin:n {tag=text}
4060     \tag_mc_begin:n {tag=text}
4061 }
4062 \socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {stop-list-tags}
4063 {
4064     \tag_mc_end:
4065     \tag_struct_end:n {tag=text}
4066     \tag_struct_end:n {tag=text-unit}
4067     \tag_struct_end:n {tag=LBody}
4068     \tag_struct_end:n {tag=LI}
4069     \tag_mc_begin_pop:n {}
4070     \tag_suspend:n {keyanspic}
4071 }

```

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

```

4072 \cs_new_protected_nopar:Nn \__enumext_anspic_start_list_tag:
4073 {
4074     \IfDocumentMetadataTF
4075     {
4076         \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {start-list-tags}
4077         \socket_use:n {tagsupport/__enumext/keyanspic}
4078     } {}
4079 }
4080 \cs_new_protected_nopar:Nn \__enumext_anspic_stop_start_list_tag:
4081 {
4082     \IfDocumentMetadataTF
4083     {
4084         \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {stop-start-tags}
4085         \socket_use:n {tagsupport/__enumext/keyanspic}
4086     } {}

```

```

4087     }
4088     \cs_new_protected_nopar:Nn \__enumext_anspic_stop_list_tag:
4089     {
4090         \IfDocumentMetadataTF
4091         {
4092             \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {stop-list-tags}
4093             \socket_use:n {tagsupport/__enumext/keyanspic}
4094         } {}
4095     }

```

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

13.42 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 as replacement for `\item` command and placed inside `minipage` environments, with the `<label>` centered “*above*” or “*below*”, adjusting *widths* and *position* according to the options passed to the environment.

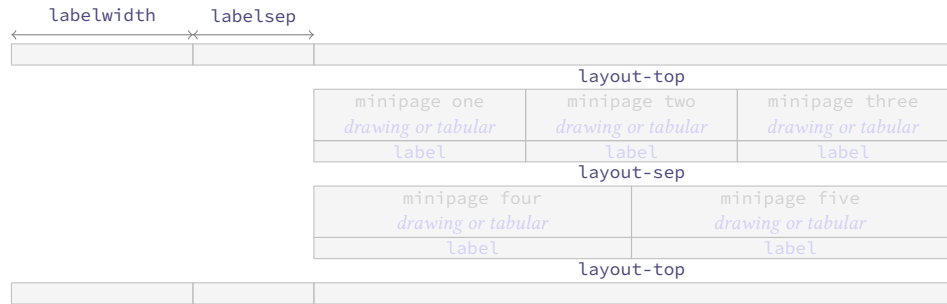


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

In order for the `keyanspic` environment and the `\anspic` command to work correctly, we need to set and export some variables in the first part of the environment definition and pass them to `\anspic` which is executed in the second part of the environment. This implementation is adapted from the answer given by Enrico Gregorio (@egreg) in [How to process the body of an environment and divide it by a \macro?](#).

13.42.1 The environment `keyanspic`

First we define the key that allows us to process the position of the `<label>` centered “*above*” or “*below*” which will be `label-pos`, the vertical separation of these from *drawing or tabular* will be handled with the key `label-sep`. The “*layout style*” will be handled with the key `layout-sty` will take two values separated by comma `{(n° upper, n° lower)}` and will determine the number of `minipage` environments in which all arguments of `\anspic` will be printed at the “upper” and “lower” within the environments separated by the value of the key `layout-sep`. The vertical space “top” and “bottom” of the environment will be handled with the key `layout-top`.

```

label-pos 231 First we define the key that allows us to process the position of the <label> centered “above” or “below” which
label-sep 231 will be label-pos, the vertical separation of these from drawing or tabular will be handled with the key
layout-sty 231 label-sep. The “layout style” will be handled with the key layout-sty will take two values separated
layout-sep 231 by comma {(n° upper, n° lower)} and will determine the number of minipage environments in which all
layout-top 231 arguments of \anspic will be printed at the “upper” and “lower” within the environments separated by the
mark-ans 231 value of the key layout-sep. The vertical space “top” and “bottom” of the environment will be handled with
mark-pos 231 the key layout-top.
mark-sep 231
save-sep 231
wrap-opt 231
wrap-key 231
show-ans 231
show-pos 231

4096 \keys_define:nn { enumext / keyanspic }
4097 {
4098     label-pos .choice:,
4099     label-pos / above .code:n =
4100         \bool_set_true:N \l__enumext_anspic_label_above_bool
4101         \str_set:Nn \l__enumext_anspic_mini_pos_str { t },
4102     label-pos / below .code:n =
4103         \bool_set_false:N \l__enumext_anspic_label_above_bool
4104         \str_set:Nn \l__enumext_anspic_mini_pos_str { b },
4105     label-pos / unknown .code:n =
4106         \msg_error:nnee { enumext } { unknown-choice }
4107         { label-pos } { above,~ below } { \exp_not:n {#1} },
4108     label-pos .initial:n = below,
4109     label-pos .value_required:n = true,
4110     label-sep .skip_set:N = \l__enumext_anspic_label_sep_skip,
4111     label-sep .value_required:n = true,
4112     layout-sty .tl_set:N = \l__enumext_anspic_layout_style_tl,
4113     layout-sty .value_required:n = true,
4114     layout-sep .code:n = \keys_set:nn { enumext / keyans } { parsep = #1 },
4115     layout-sep .value_required:n = true,
4116     layout-top .code:n = \keys_set:nn { enumext / keyans } { topsep = #1 },
4117     layout-top .value_required:n = true,
4118     mark-ans .code:n = \keys_set:nn { enumext / keyans } { mark-ans = #1 },
4119     mark-ans .value_required:n = true,
4120     mark-pos .code:n = \keys_set:nn { enumext / keyans } { mark-pos = #1 },
4121     mark-pos .value_required:n = true,

```

```

4122 mark-sep .code:n = \keys_set:nn { enumext / keyans } { mark-sep = #1 },
4123 mark-sep .value_required:n = true,
4124 save-sep .code:n = \keys_set:nn { enumext / keyans } { save-sep = #1 },
4125 save-sep .value_required:n = true,
4126 wrap-opt .code:n = \keys_set:nn { enumext / keyans } { wrap-opt = #1 },
4127 wrap-opt .value_required:n = true,
4128 wrap-key .code:n = \keys_set:nn { enumext / keyans } { wrap-key = #1 },
4129 wrap-key .value_required:n = true,
4130 show-ans .code:n = \keys_set:nn { enumext / keyans } { show-ans = #1 },
4131 show-ans .value_required:n = true,
4132 show-pos .code:n = \keys_set:nn { enumext / keyans } { show-pos = #1 },
4133 show-pos .value_required:n = true,
4134 unknown .code:n = { \__enumext_keyans_unknown_keys:n {#1} }
4135 }

```

(End of definition for label-pos and others.)

```

\__enumext_keyans_pic_safe_exec:
\__enumext_keyans_pic_parse_keys:n
\__enumext_keyans_pic_skip_abs:N
\__enumext_keyans_pic_arg_two:

```

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

```

4136 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
4137 {
4138   \int_incr:N \__enumext_keyans_pic_level_int
4139   \int_compare:nNnT { \__enumext_keyans_pic_level_int } > { 1 }
4140   {
4141     \msg_error:nn { enumext } { keyanspic-nested }
4142   }
4143   \__enumext_keyans_name_and_start:
4144 }

```

Parse [`key = val`] for `keyanspic` environment.

```

4145 \cs_new_protected:Npn \__enumext_keyans_pic_parse_keys:n #1
4146 {
4147   \tl_if_novalue:nF {#1}
4148   {
4149     \keys_set:nn { enumext / keyanspic } {#1}
4150   }
4151 }

```

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

```

4152 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
4153 {
4154   \dim_compare:nNnT { #1 } < { \c_zero_dim }
4155   {
4156     \skip_set:Nn #1 { -#1 }
4157   }
4158 }

```

The `__enumext_keyans_pic_arg_two:` function will be used in the *second argument* of the `list` environment that defines the `keyanspic` environment, with this we will take the configuration of the “spaces” and the keys `label`, `wrap-label`, `parsep` and `topsep` from the `keyans` environment. The first thing we need to do is set the boolean variable `__enumext_leftmargin_tmp_v_bool` handled by the `list-indent` key to “false”, then copy the definition of the second list argument from the `keyans` environment definition and make sure that `\parsep` does not have a negative value.

```

4159 \cs_new_protected:Npn \__enumext_keyans_pic_arg_two:
4160 {
4161   \bool_set_false:N \__enumext_leftmargin_tmp_v_bool
4162   \__enumext_list_arg_two_v:
4163   \__enumext_keyans_pic_skip_abs:N \parsep

```

Now we increment the counter `enumXv` of the `keyans` environment and save the *total height* of the `(label)` in `__enumext_anspic_label_htdp_dim` used by `\anspic` and we will adjust the values of `\parsep` only if the key `label-pos` is set to *below*.

```

4164   \bool_if:NF \__enumext_anspic_label_above_bool
4165   {
4166     \stepcounter { enumXv }
4167     \hbox_set:Nn \__enumext_anspic_label_box { \__enumext_label_v_tl }
4168     \dim_set:Nn \__enumext_anspic_label_htdp_dim
4169     {
4170       \box_ht_plus_dp:N \__enumext_anspic_label_box
4171     }
4172     \skip_add:Nn \parsep

```

```

4173     {
4174         \l__enumext_anspic_label_htdp_dim
4175         + \box_dp:N \strutbox
4176         + \l__enumext_anspic_label_sep_skip
4177     }
4178 }

```

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

```

4179 \dim_add:Nn \leftmargin { -\l__enumext_labelwidth_v_dim - \l__enumext_labelsep_v_dim }
4180 \ignorespaces
4181 \skip_add:Nn \topsep { 0.5\box_dp:N \strutbox }
4182 \dim_zero:N \listparindent
4183 \skip_zero:N \partopsep
4184 \skip_zero:N \itemsep
4185 }

```

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

keyanspic Now we define the environment `keyanspic`. For compatibility with *tagged* PDF we must use the `\begin{list}` form and a lot of conditional code using `\IfDocumentMetadataTF`. We will first stop the code for automatic *tagged* PDF for `list` environments, redefine `\item` so that it cannot be used, and stop the code for automatic *tagged* PDF for the `keyanspic` environment.

```

4186 \NewDocumentEnvironment{keyanspic}{o}
4187 {
4188     \__enumext_keyans_pic_safe_exec:
4189     \__enumext_keyans_pic_parse_keys:n {#1}
4190     \begin{list} {} { \__enumext_keyans_pic_arg_two: }
4191     \IfDocumentMetadataTF
4192     {
4193         \tag_suspend:n {list}
4194     }{}
4195     \item[] \scan_stop:
4196     \RenewDocumentCommand \item {}
4197     {
4198         \msg_error:nn { enumext } { keyanspic-item-cmd }
4199     }
4200     \IfDocumentMetadataTF
4201     {
4202         \tag_resume:n {keyanspic}
4203         \tag_tool:n {para/tagging=false}
4204         \tag_suspend:n {keyanspic}
4205     } { }
4206 }

```

In the second part of the environment definition we will manually place our code for *tagged* PDF and execute the command `\anspic` using the `__enumext_anspic_exec`: function.

```

4207 {
4208     \IfDocumentMetadataTF
4209     {
4210         \tag_resume:n {keyanspic}
4211         \tag_mc_end_push:
4212         \tag_struct_begin:n {tag=L,attribute=enumerate}
4213     } { }
4214     \__enumext_anspic_exec:
4215     \IfDocumentMetadataTF
4216     {
4217         \tag_suspend:n {keyanspic}
4218     } { }
4219     \end{list}
4220     \IfDocumentMetadataTF
4221     {
4222         \tag_struct_end:n {tag=L}
4223         \tag_mc_begin_pop:n {}
4224         \tag_struct_end:n {tag=L}
4225         \tag_mc_begin_pop:n {}
4226     } { }

```

Finally we check if `\anspic*` has been used, set the counter `enumXvi` to zero and apply our “adjusted” vertical space bottom.

```

4227 \__enumext_check_starred_cmd:n { anspic }
4228 \setcounter { enumXvi } { 0 }

```

```

4229 \bool_if:NTF \l__enumext_anspic_label_above_bool
4230 {
4231   \par\addvspace{ 0.5\box_dp:N \strutbox }
4232 }
4233 {
4234   \par
4235   \addvspace
4236   {
4237     \dim_eval:n
4238     {
4239       \l__enumext_anspic_label_htdp_dim + \box_ht_plus_dp:N \strutbox
4240       + \l__enumext_anspic_label_sep_skip + \l__enumext_topsep_v_skip
4241     }
4242   }
4243 }
4244 }

```

(End of definition for `keyanspic`. This function is documented on page 16.)

13.42.2 The command `\anspic`

The `\anspic` command take three arguments, the *starred versions* `\anspic*[\langle content \rangle]` store the current `\label` next to the *optional argument* `[\langle content \rangle]` in the *sequence* and *prop list* defined by `save-ans` key. The third *mandatory argument* `{\langle drawing or tabular \rangle}` is NOT stored in the *sequence* or *prop list*.

- One of the complications here to make the `keyanspic` environment compatible with *tagged* PDF is the position of `\label`, the `\anspic` command processes the arguments in order, where #1 and #2 correspond to `\label` and #3 to the mandatory argument and puts all this inside a `minipage` environment. If #1 and #2, that is `\label`, is above #3 there are no problems with *tagged* PDF, but if #3 comes first the list created with *tagged* PDF will not be correct.

`\anspic`

We check that the command is active in the `keyanspic` environment only if the `save-ans` key is present, otherwise we return an error. The three arguments are handled by the function `__enumext_anspic_args:nnn` and stored in the sequence `__enumext_anspic_args_seq` which is processed by the `keyanspic` environment.

```

4245 \NewDocumentCommand \anspic { s o +m }
4246 {
4247   \bool_if:NF \l__enumext_store_active_bool
4248   {
4249     \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
4250   }
4251   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
4252   {
4253     \msg_error:nn { enumext } { keyanspic-wrong-level }
4254   }
4255   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
4256   {
4257     \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
4258   }
4259   \seq_put_right:Nn \l__enumext_anspic_args_seq
4260   {
4261     \__enumext_anspic_args:nnn { #1 } { #2 } { #3 }
4262   }
4263 }

```

The `__enumext_anspic_body_dim:n` function will set the value of `\l__enumext_anspic_body_htdp_dim` equal to the “height plus depth” of the *mandatory argument* if the key `label-pos` is set “below”.

```

4264 \cs_new_protected:Npn \__enumext_anspic_body_dim:n #1
4265 {
4266   \bool_if:NF \l__enumext_anspic_label_above_bool
4267   {
4268     \IfDocumentMetadataTF
4269     {
4270       \tag_suspend:n {keyanspic}
4271     } { }
4272     \vbox_set:Nn \l__enumext_anspic_body_box { #1 }
4273     \dim_set:Nn \l__enumext_anspic_body_htdp_dim
4274     {
4275       \box_ht_plus_dp:N \l__enumext_anspic_body_box
4276     }
4277     \IfDocumentMetadataTF
4278     {
4279       \tag_resume:n {keyanspic}
4280     } { }

```



```

4281     }
4282 }

```

The `__enumext_anspic_label:nn` function will process inside `\makebox` the *starred argument* ‘*’ and *optional argument* passed to the command. Here we will store the `<label>` and *optional argument* in *prop list* and *sequence* and execute the `show-ans`, `show-pos`, `font`, `wrap-label`, `wrap-key` and `wrap-opt` keys.

```

4283 \cs_new_protected:Npn \__enumext_anspic_label:nn #1 #2
4284 {
4285   \makebox[ \l__enumext_anspic_mini_width_dim ][ c ]
4286   {
4287     \bool_if:NTF { #1 }
4288     {
4289       \bool_set_true:N \l__enumext_item_wrap_key_bool
4290       \bool_set_true:N \l__enumext_wrap_label_v_bool
4291       \__enumext_keyans_save_item_opt:n { #2 }
4292       \__enumext_keyans_addto_prop:n { #2 }
4293       \__enumext_keyans_store_ref:
4294       \__enumext_keyans_addto_seq:n { #2 }
4295       \int_gincr:N \g__enumext_check_starred_cmd_int
4296       \__enumext_keyans_show_ans:
4297       \__enumext_keyans_show_pos:
4298       \tl_use:N \l__enumext_label_font_style_v_tl
4299       \__enumext_keyans_wrapper_label:n { \l__enumext_label_vi_tl }
4300       \c_space_tl \__enumext_keyans_show_item_opt:
4301     }
4302     {
4303       \bool_set_false:N \l__enumext_item_wrap_key_bool
4304       \tl_use:N \l__enumext_label_font_style_v_tl
4305       \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl }
4306     }
4307   }
4308 }

```

The function `__enumext_anspic_label_pos:nnn` will be in charge of handling the “*counter*” and the position of the `<label>`, set by `label-pos` key which will have the same configuration as the `keyans` environment.

```

4309 \cs_new_protected:Npn \__enumext_anspic_label_pos:nnn #1 #2 #3
4310 {
4311   \stepcounter { enumXvi }
4312   \__enumext_anspic_body_dim:n { #3 }
4313   \bool_if:NTF \l__enumext_anspic_label_above_bool
4314   {
4315     \__enumext_anspic_label:nn { #1 } { #2 }
4316   }
4317   {
4318     \raisebox
4319     {
4320       -\dim_eval:n
4321       {
4322         \l__enumext_anspic_label_htdp_dim
4323         + \l__enumext_anspic_body_htdp_dim
4324         + \box_dp:N \strutbox
4325         + \l__enumext_anspic_label_sep_skip
4326       }
4327     }
4328     [ opt ] [ opt ]
4329     {
4330       \__enumext_anspic_label:nn { #1 } { #2 }
4331     }
4332   }
4333 }
4334 %

```

The `__enumext_anspic_args:nnn` function will be responsible for placing the code compatible with *tagged* PDF and the arguments within the `\l__enumext_anspic_args_seq` sequence which will be processed by the `__enumext_anspic_print:n` function in the second part of the definition of the `keyanspic` environment.

```

4335 \cs_new_protected:Nn \__enumext_anspic_args:nnn
4336 {
4337   \__enumext_anspic_start_list_tag:
4338   \__enumext_anspic_label_pos:nnn { #1 } { #2 } { #3 }
4339   \__enumext_anspic_stop_start_list_tag:
4340   \bool_if:NTF \l__enumext_anspic_label_above_bool
4341   {

```

```

4342     \[\l__enumext_anspic_label_sep_skip] #3
4343   }
4344   {
4345     \[ #3
4346   }
4347   \__enumext_anspic_stop_list_tag:
4348 }

```

The value $\langle n^{\circ upper}, n^{\circ lower} \rangle$ passed to the `layout-sty` key is split by comma and is handled directly by the function `__enumext_anspic_print:n` and passed to the function `__enumext_anspic_row:n`.

```

4349 \cs_new_protected:Nn \__enumext_anspic_print:n
4350 {
4351   \clist_map_function:nN { #1 } \__enumext_anspic_row:n
4352 }
4353 \cs_generate_variant:Nn \__enumext_anspic_print:n { e, V }

```

The function `__enumext_anspic_row:n` will set the *widths* for the `minipage` environments and place *all arguments* passed to `\anspic` saved in the `\l__enumext_anspic_args_seq` sequence inside them.

```

4354 \cs_new_protected:Nn \__enumext_anspic_row:n
4355 {
4356   \dim_set:Nn \l__enumext_anspic_mini_width_dim { \linewidth / #1 }
4357   \int_set:Nn \l__enumext_anspic_above_int { \l__enumext_anspic_below_int }
4358   \int_set:Nn \l__enumext_anspic_below_int { \l__enumext_anspic_above_int + #1 }
4359   \int_step_inline:nnn
4360     { \l__enumext_anspic_above_int + 1 }
4361     { \l__enumext_anspic_below_int }
4362     {
4363       \IfDocumentMetadataTF
4364       {
4365         \tag_suspend:n {minipage}
4366       } { }
4367       \begin{minipage}[ \l__enumext_anspic_mini_pos_str ]{ \l__enumext_anspic_mini_width_dim }
4368         \centering
4369         \seq_item:Nn \l__enumext_anspic_args_seq { ##1 }
4370       \end{minipage}
4371       \IfDocumentMetadataTF
4372       {
4373         \tag_resume:n {minipage}
4374       } { }
4375     }
4376   \par
4377 }

```

The `__enumext_anspic_exec:` function will execute all the code in the `\anspic` command in the second argument of the `keyanspic` environment definition. If the key `layout-sty` is not set, everything will be printed on a *single line*.

```

4378 \cs_new_protected:Nn \__enumext_anspic_exec:
4379 {
4380   \tl_if_empty:NTF \l__enumext_anspic_layout_style_tl
4381   {
4382     \__enumext_anspic_print:e { \seq_count:N \l__enumext_anspic_args_seq }
4383   }
4384   {
4385     \__enumext_anspic_print:V \l__enumext_anspic_layout_style_tl
4386   }
4387 }

```

(End of definition for `\anspic` and others. This function is documented on page 16.)

13.43 The horizontal environments

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

To achieve the *horizontal list environments* we will capture the `\item` command and the $\langle content \rangle$ of this in *horizontal box* using `\makebox` for the `label` and a `minipage` environment for the $\langle content \rangle$ passed to `\item`, we will also add the *optional argument* $\langle number \rangle$ to `\item` to be able to *join columns* horizontally, in simple terms, we want `\item` to behave in the same way as in the `enumext` environment but adding an *first optional argument* $\langle number \rangle$.

A side effect is the limitation of using `\item` in this way *without* using `\RenewDocumentCommand`, which loses the original definition and affects the *standard list environments* provided by \LaTeX and any environment defined

using base `list` environment, including: `itemize`, `enumerate`, `description`, `quote`, `quotation`, `verse`, `center`, `flushleft`, `flushright`, `verbatim`, `tabbing`, `trivlist`, `list` and all environments created with `\newtheorem`.

One way to get around this is to use something like:

```
\AddToHook{env/enumerate/before}{recover original \item definition}
```

inside `minipage`, but in my partial tests this does not have the desired effect and the vertical and horizontal spacing is distorted. For now this will remain as a limitation and I will see if it is feasible to implement it in the future.

For compatibility with the *tagged* PDF we close the environments according to the presence or not of the `mini-env` key.

13.43.1 Functions for item box width

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:
4388 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
4389 {
4390   \dim_compare:nNnT { \__enumext_columns_sep_vii_dim } = { \c_zero_dim }
4391   {
4392     \dim_set:Nn \__enumext_columns_sep_vii_dim
4393     {
4394       ( \__enumext_labelwidth_vii_dim + \__enumext_labelsep_vii_dim )
4395       / \__enumext_columns_vii_int
4396     }
4397   }
4398   \int_set:Nn \__enumext_tmpa_vii_int { \__enumext_columns_vii_int - 1 }
4399   \dim_set:Nn \__enumext_item_width_vii_dim
4400   {
4401     ( \linewidth - \__enumext_columns_sep_vii_dim * \__enumext_tmpa_vii_int )
4402     / \__enumext_columns_vii_int
4403     - \__enumext_labelwidth_vii_dim
4404     - \__enumext_labelsep_vii_dim
4405   }
```

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

```
4406   \dim_compare:nNnT { \__enumext_rightmargin_vii_dim } > { \c_zero_dim }
4407   {
4408     \dim_sub:Nn \__enumext_item_width_vii_dim
4409     {
4410       ( \__enumext_rightmargin_vii_dim * \__enumext_tmpa_vii_int )
4411       / \__enumext_columns_vii_int
4412     }
4413     \dim_add:Nn \__enumext_columns_sep_vii_dim
4414     {
4415       \__enumext_rightmargin_vii_dim
4416     }
4417   }
4418 }
```

Same implementation for the `keyans*` environment.

```
4419 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
4420 {
4421   \dim_compare:nNnT { \__enumext_columns_sep_viii_dim } = { \c_zero_dim }
4422   {
4423     \dim_set:Nn \__enumext_columns_sep_viii_dim
4424     {
4425       ( \__enumext_labelwidth_viii_dim + \__enumext_labelsep_viii_dim )
4426       / \__enumext_columns_viii_int
4427     }
4428   }
4429   \int_set:Nn \__enumext_tmpa_viii_int { \__enumext_columns_viii_int - 1 }
4430   \dim_set:Nn \__enumext_item_width_viii_dim
4431   {
4432     ( \linewidth - \__enumext_columns_sep_viii_dim * \__enumext_tmpa_viii_int )
4433     / \__enumext_columns_viii_int
4434     - \__enumext_labelwidth_viii_dim
4435     - \__enumext_labelsep_viii_dim
4436   }
4437   \dim_compare:nNnT { \__enumext_rightmargin_viii_dim } > { \c_zero_dim }
4438   {
4439     \dim_sub:Nn \__enumext_item_width_viii_dim
4440     {
4441       ( \__enumext_rightmargin_viii_dim * \__enumext_tmpa_viii_int )
4442       / \__enumext_columns_viii_int
4443     }
4444   }
```

```

4443     }
4444     \dim_add:Nn \l__enumext_columns_sep_viii_dim
4445     {
4446         \l__enumext_rightmargin_viii_dim
4447     }
4448 }
4449 }

```

(End of definition for `__enumext_starred_columns_set_vii:` and `__enumext_starred_columns_set_viii:`)

13.43.2 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.

```

4450 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
4451 {
4452     \int_set:Nn \l__enumext_joined_item_vii_int {#1}
4453     \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
4454     {
4455         \msg_warning:nnee { enumext } { item-joined }
4456         { \int_use:N \l__enumext_joined_item_vii_int }
4457         { \int_use:N \l__enumext_columns_vii_int }
4458         \int_set:Nn \l__enumext_joined_item_vii_int
4459         {
4460             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4461         }
4462     }
4463     \int_compare:nNnT
4464     { \l__enumext_joined_item_vii_int }
4465     >
4466     { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4467     {
4468         \msg_warning:nnee { enumext } { item-joined-columns }
4469         { \int_use:N \l__enumext_joined_item_vii_int }
4470         {
4471             \int_eval:n
4472             { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4473         }
4474         \int_set:Nn \l__enumext_joined_item_vii_int
4475         {
4476             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4477         }
4478     }
4479     \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { 1 }
4480     {
4481         \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
4482         \int_decr:N \l__enumext_joined_item_aux_vii_int
4483         \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
4484         \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
4485         \dim_set:Nn \l__enumext_joined_width_vii_dim
4486         {
4487             \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
4488             + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
4489                 + \l__enumext_columns_sep_vii_dim
4490             ) * \l__enumext_joined_item_aux_vii_int
4491         }
4492         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
4493     }
4494     {
4495         \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
4496         \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
4497     }
4498 }

```

Same implementation for the `keyans*` environment.

```

4499 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
4500 {
4501     \int_set:Nn \l__enumext_joined_item_viii_int {#1}
4502     \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
4503     {
4504         \msg_warning:nnee { enumext } { item-joined }

```

```

4505         { \int_use:N \l__enumext_joined_item_viii_int }
4506         { \int_use:N \l__enumext_columns_viii_int }
4507     \int_set:Nn \l__enumext_joined_item_viii_int
4508     {
4509         \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4510     }
4511 }
4512 \int_compare:nNnT
4513 { \l__enumext_joined_item_viii_int }
4514 >
4515 { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4516 {
4517     \msg_warning:nnee { enumext } { item-joined-columns }
4518     { \int_use:N \l__enumext_joined_item_viii_int }
4519     {
4520         \int_eval:n
4521         { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4522     }
4523     \int_set:Nn \l__enumext_joined_item_viii_int
4524     {
4525         \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4526     }
4527 }
4528 \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { 1 }
4529 {
4530     \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
4531     \int_decr:N \l__enumext_joined_item_aux_viii_int
4532     \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
4533     \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
4534     \dim_set:Nn \l__enumext_joined_width_viii_dim
4535     {
4536         \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
4537         + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
4538           + \l__enumext_columns_sep_viii_dim
4539         ) * \l__enumext_joined_item_aux_viii_int
4540     }
4541     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
4542 }
4543 {
4544     \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
4545     \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
4546 }
4547 }

```

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

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

```

\__enumext_start_mini_vii:
\__enumext_stop_mini_vii:

```

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

```

4548 \cs_new_protected:Nn \__enumext_start_mini_vii:
4549 {
4550     \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
4551     {
4552         \dim_set:Nn \l__enumext_minipage_left_vii_dim
4553         {
4554             \linewidth
4555             - \l__enumext_minipage_right_vii_dim
4556             - \l__enumext_minipage_hsep_vii_dim
4557         }
4558         \bool_set_true:N \l__enumext_minipage_active_vii_bool
4559         \dim_gset_eq:NN
4560         \g__enumext_minipage_right_vii_dim
4561         \l__enumext_minipage_right_vii_dim
4562         \__enumext_mini_addvspace_vii:
4563         \nointerlineskip\nindent
4564         \__enumext_mini_page{ \l__enumext_minipage_left_vii_dim }
4565     }
4566 }

```

The function `__enumext_stop_mini_vii:` closes the `__enumext_mini_page` environment on the “*left side*”, applies `\hfill` and set the variable `\g__enumext_minipage_active_vii_bool` to “*true*” which will be used in the function `__enumext_after_env:nn` to execute the `minipage` on the “*right side*”. At this point we will execute the `__enumext_stop_list:` and `__enumext_stop_store_level_vii:` functions stopping the `list` environment and the level saving mechanism for storage in *sequence* of the `\anskey` command and `anskey*` environment. This function is passed to the `__enumext_after_list_vii:` function in the second part of the `enumext*` environment definition (§13.44).

```

4567 \cs_new_protected:Nn \__enumext_stop_mini_vii:
4568 {
4569   \bool_if:NTF \l__enumext_minipage_active_vii_bool
4570   {
4571     \__enumext_stop_list:
4572     \__enumext_stop_store_level_vii:
4573     \IfDocumentMetadataTF { \tag_resume:n {enumext*} } { }
4574     \end__enumext_mini_page
4575     \hfill
4576     \bool_gset_true:N \g__enumext_minipage_active_vii_bool
4577   }
4578   {
4579     \__enumext_stop_list:
4580     \__enumext_stop_store_level_vii:
4581   }
4582 }

```

(End of definition for `__enumext_start_mini_vii:` and `__enumext_stop_mini_vii:`.)

Finally we execute the `{\code}` passed to the `mini-right` or `mini-right*` keys stored in the variable `\g__enumext_miniright_code_vii_tl` in the `minipage` environment on the “*right side*”. For compatibility with the `caption` package and possibly other `{\code}` passed to this key, we will pass it to a box and then print it.

```

4583 \__enumext_after_env:nn {enumext*}
4584 {
4585   \bool_if:NT \g__enumext_minipage_active_vii_bool
4586   {
4587     \__enumext_minipage:w [ t ] { \g__enumext_minipage_right_vii_dim }
4588     \legacy_if_gset_false:n { @minipage }
4589     \skip_vertical:N \c_zero_skip
4590     \par\addvspace { \g__enumext_minipage_right_skip }
4591     \bool_if:NF \g__enumext_minipage_center_vii_bool
4592     {
4593       \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
4594       {
4595         \centering
4596       }
4597     }
4598     \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
4599     {
4600       \tl_use:N \g__enumext_miniright_code_vii_tl
4601     }
4602     \box_use_drop:N \l__enumext_miniright_code_vii_box
4603     \skip_vertical:N \c_zero_skip
4604     \__enumext_endminipage:
4605     \par\addvspace{ \g__enumext_minipage_after_skip }
4606   }
4607   \bool_gset_false:N \g__enumext_minipage_active_vii_bool
4608   \bool_gset_true:N \g__enumext_minipage_center_vii_bool
4609   \tl_gclear:N \g__enumext_miniright_code_vii_tl
4610   \dim_gzero:N \g__enumext_minipage_right_vii_dim
4611   \bool_gset_false:N \g__enumext_starred_bool
4612 }

```

`__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.

```

4613 \cs_new_protected:Nn \__enumext_start_mini_viii:
4614 {
4615   \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
4616   {
4617     \dim_set:Nn \l__enumext_minipage_left_viii_dim
4618     {
4619       \linewidth

```

```

4620         - \l__enumext_minipage_right_viii_dim
4621         - \l__enumext_minipage_hsep_viii_dim
4622     }
4623     \bool_set_true:N \l__enumext_minipage_active_viii_bool
4624     \dim_gset_eq:NN
4625         \g__enumext_minipage_right_viii_dim
4626         \l__enumext_minipage_right_viii_dim
4627     \__enumext_mini_addvspace_viii:
4628     \nointerlineskip\noindent
4629     \__enumext_mini_page{ \l__enumext_minipage_left_viii_dim }
4630 }
4631 }
4632 \cs_new_protected:Nn \__enumext_stop_mini_viii:
4633 {
4634     \bool_if:NTF \l__enumext_minipage_active_viii_bool
4635     {
4636         \__enumext_stop_list:
4637         \IfDocumentMetadataTF { \tag_resume:n {keyans*} } { }
4638         \end__enumext_mini_page
4639         \hfill
4640         \bool_gset_true:N \g__enumext_minipage_active_viii_bool
4641     }
4642     {
4643         \__enumext_stop_list:
4644     }
4645 }
4646 \__enumext_after_env:nn {keyans*}
4647 {
4648     \bool_if:NT \g__enumext_minipage_active_viii_bool
4649     {
4650         \__enumext_mini_page{ \g__enumext_minipage_right_viii_dim }
4651         \par\addvspace { \g__enumext_minipage_right_skip }
4652         \bool_if:NF \g__enumext_minipage_center_viii_bool
4653         {
4654             \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
4655                 {
4656                     \centering
4657                 }
4658         }
4659         \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
4660         {
4661             \tl_use:N \g__enumext_miniright_code_viii_tl
4662         }
4663         \box_use_drop:N \l__enumext_miniright_code_viii_box
4664         \end__enumext_mini_page
4665         \par\addvspace{ \g__enumext_minipage_after_skip }
4666     }
4667     \bool_gset_false:N \g__enumext_minipage_active_viii_bool
4668     \bool_gset_true:N \g__enumext_minipage_center_viii_bool
4669     \tl_gclear:N \g__enumext_miniright_code_viii_tl
4670     \dim_gzero:N \g__enumext_minipage_right_viii_dim
4671 }

```

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

13.44 The environment enumext*

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

```

4672 \NewDocumentEnvironment{enumext*}{ o }
4673 {
4674     \__enumext_safe_exec_vii:
4675     \__enumext_parse_keys_vii:n {#1}
4676     \__enumext_before_list_vii:
4677     \__enumext_start_store_level_vii:
4678     \__enumext_start_list:nn { }
4679     {

```



```

4680     \__enumext_list_arg_two_vii:
4681     \__enumext_before_keys_exec_vii:
4682   }
4683   \IfDocumentMetadataTF { \tag_suspend:n {enumext*} } { }
4684   \__enumext_starred_columns_set_vii:
4685   \item[] \scan_stop:
4686   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_first_item_tmp_vii:
4687   \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
4688   \ignorespaces
4689 }
4690 {
4691   \IfDocumentMetadataTF { \tag_struct_end:n {tag=text-unit} } { }
4692   \__enumext_stop_item_tmp_vii:
4693   \__enumext_remove_extra_parsep_vii:
4694   \__enumext_after_list_vii:
4695 }

```

(End of definition for `enumext*`. This function is documented on page 5.)

`__enumext_safe_exec_vii:` We will first call the function `__enumext_is_not_nested:` which sets `\g__enumext_starred_bool` to true if we are NOT nested within `enumext`, then call the function `__enumext_internal_mini_page:` to create the environment `__enumext_mini_page`, 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.

```

4696 \cs_new_protected:Nn \__enumext_safe_exec_vii:
4697 {
4698   \__enumext_is_not_nested:
4699   \__enumext_internal_mini_page:
4700   \int_incr:N \l__enumext_level_h_int
4701   \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
4702   {
4703     \msg_error:nn { enumext } { nested }
4704   }
4705   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
4706   {
4707     \msg_error:nnn { enumext } { nested-horizontal } { keyans*}
4708   }
4709   \bool_set_true:N \l__enumext_starred_bool
4710   \bool_set_false:N \l__enumext_standar_bool
4711   \__enumext_is_on_first_level:
4712 }

```

(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.

```

4713 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
4714 {
4715   \tl_if_novalue:nF {#1}
4716   {
4717     \str_clear:N \l__enumext_series_str
4718     \keys_set:nn { enumext / enumext* } {#1}
4719     \__enumext_parse_series:n {#1}
4720     \__enumext_store_active_keys_vii:n {#1}
4721   }
4722 }

```

(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*`.

```

4723 \cs_new_protected:Nn \__enumext_before_list_vii:
4724 {
4725   \__enumext_vspace_above_vii:
4726   \__enumext_check_ans_active:

```

```

4727     \__enumext_before_args_exec_vii:
4728     \__enumext_start_mini_vii:
4729 }

```

(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: which internally calls __enumext_stop_list: and __enumext_stop_store_level_vii: (§13.43.3) 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.

```

4730 \cs_new_protected:Nn \__enumext_after_list_vii:
4731 {
4732     \__enumext_stop_mini_vii:
4733     \__enumext_after_stop_list_vii:
4734     \__enumext_check_ans_key_hook:
4735     \__enumext_vspace_below_vii:
4736     \bool_set_false:N \__enumext_starred_bool
4737     \__enumext_resume_save_counter:
4738 }

```

(End of definition for __enumext_after_list_vii:.)

__enumext_start_store_level_vii:

__enumext_stop_store_level_vii:

The __enumext_start_store_level_vii: and __enumext_stop_store_level_vii: functions activate the “storing structure” mechanism in *sequence* for \anskey command and `anskey*` environment if `enumext*` are nested in `enumext`.

```

4739 \cs_new_protected:Nn \__enumext_start_store_level_vii:
4740 {
4741     \bool_if:NT \__enumext_store_active_bool
4742     {
4743         \int_compare:nNnT { \__enumext_level_int } > { 0 }
4744         {
4745             \__enumext_store_level_open_vii:
4746         }
4747     }
4748 }
4749 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
4750 {
4751     \bool_if:NT \__enumext_store_active_bool
4752     {
4753         \int_compare:nNnT { \__enumext_level_int } > { 0 }
4754         {
4755             \__enumext_store_level_close_vii:
4756         }
4757     }
4758 }

```

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

13.44.1 The command \item in enumext*

__enumext_first_item_tmp_vii:

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

```

4759 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_vii:
4760 {
4761     \skip_horizontal:n
4762     {
4763         -\__enumext_labelwidth_vii_dim - \__enumext_labelsep_vii_dim
4764     }
4765     \ignorespaces
4766 }

```

(End of definition for __enumext_first_item_tmp_vii:.)

__enumext_start_item_tmp_vii:

__enumext_item_peek_args_vii:

__enumext_joined_item_vii:w

__enumext_standar_item_vii:w

__enumext_starred_item_vii:w

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`.

```

4767 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
4768 {
4769   \__enumext_stop_item_tmp_vii:
4770   \int_incr:N \l__enumext_item_column_pos_vii_int
4771   \int_gincr:N \g__enumext_item_count_all_vii_int
4772   \__enumext_item_peek_args_vii:
4773 }

```

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).

```

4774 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
4775 {
4776   \peek_meaning:NTF (
4777     { \__enumext_joined_item_vii:w }
4778     { \__enumext_joined_item_vii:w (1) }
4779   }

```

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`.

```

4780 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
4781 {
4782   \__enumext_starred_joined_item_vii:n {#1}
4783   \peek_meaning_remove:NTF *
4784     { \__enumext_starred_item_vii:w }
4785     { \__enumext_standar_item_vii:w }
4786 }

```

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]`.

```

4787 \cs_new_protected:Npn \__enumext_standar_item_vii:w
4788 {
4789   \bool_set_false:N \l__enumext_item_starred_vii_bool
4790   \peek_meaning:NTF [
4791     {
4792       \bool_set_eq:NN \l__enumext_wrap_label_vii_bool \l__enumext_wrap_label_opt_vii_bool
4793       \__enumext_start_item_vii:w
4794     }
4795     {
4796       \bool_set_true:N \l__enumext_wrap_label_vii_bool
4797       \legacy_if_set_true:n { @noitemarg }
4798       \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ] \ignorespaces
4799     }
4800   }

```

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>]`.

```

4801 \cs_new_protected:Npn \__enumext_starred_item_vii:w
4802 {
4803   \bool_set_true:N \l__enumext_item_starred_vii_bool
4804   \bool_set_true:N \l__enumext_wrap_label_vii_bool
4805   \peek_meaning:NTF [
4806     { \__enumext_starred_item_vii_aux_i:w }
4807     { \__enumext_starred_item_vii_aux_ii:w }
4808   }
4809   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
4810   {
4811     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
4812     \__enumext_starred_item_vii_aux_ii:w
4813   }
4814   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
4815   {

```

```

4816 \peek_meaning:NTF [
4817 { \__enumext_starred_item_vii_aux_iii:w }
4818 {
4819 \dim_set_eq:NN \l__enumext_item_symbol_sep_vii_dim \l__enumext_labelsep_vii_dim
4820 \legacy_if_set_true:n { @noitemarg }
4821 \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ] \ignorespaces
4822 }
4823 }
4824 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
4825 {
4826 \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
4827 \legacy_if_set_true:n { @noitemarg }
4828 \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ] \ignorespaces
4829 }

```

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

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

- For compatibility with *tagged* PDF and `hyperref` when an environment `enumext` is nested in `enumext*` and the key `save-ans` is not active need setting the `\if@hyper@item` switch to “true”. The explanation for this is given by the master Heiko Oberdiek on `\refstepcounter{enumi}` twice (or more) creates destination with the same identifier. This patch is only needed if you are running `pdflatex` and not if you are running `lualatex`

```

4830 \cs_new_protected_nopar:Npn \__enumext_fake_make_label_vii:n #1
4831 {
4832 \legacy_if:nT { @noitemarg }
4833 {
4834 \legacy_if_set_false:n { @noitemarg }
4835 \legacy_if:nT { @nmbrrlist }
4836 {
4837 \IfDocumentMetadataTF
4838 {
4839 \bool_if:NT \l__enumext_hyperref_bool
4840 {
4841 \legacy_if_set_true:n { @hyper@item }
4842 }
4843 } { }
4844 \refstepcounter{enumXvii}
4845 \bool_if:NT \l__enumext_check_answers_bool
4846 {
4847 \int_gincr:N \g__enumext_item_number_int
4848 \bool_set_true:N \l__enumext_item_number_bool
4849 }
4850 }
4851 }
4852 \bool_if:NT \l__enumext_item_starred_vii_bool
4853 {
4854 \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
4855 {
4856 \tl_gset_eq:NN
4857 \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
4858 }
4859 \mode_leave_vertical:
4860 \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
4861 \hbox_overlap_left:n { \g__enumext_item_symbol_aux_vii_tl }
4862 \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
4863 \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
4864 }
4865 \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
4866 {
4867 \tl_use:N \l__enumext_label_font_style_vii_tl
4868 \bool_if:NTF \l__enumext_wrap_label_vii_bool
4869 {
4870 \__enumext_wrapper_label_vii:n {#1}
4871 }

```

```

4872         { #1 }
4873     }
4874     \skip_horizontal:N \l__enumext_labelsep_vii_dim \ignorespaces
4875 }

```

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

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

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

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

```

4876 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
4877 {
4878     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
4879     \hbox_set_to_wd:Nnw \l__enumext_item_text_vii_box
4880     {
4881         \l__enumext_joined_width_vii_dim
4882         + \l__enumext_labelwidth_vii_dim
4883         + \l__enumext_labelsep_vii_dim
4884     }

```

Redefine the `\footnote` command.

```

4885     \__enumext_renew_footnote_starred:

```

Now we insert our *sockets* for *tagging* PDF support and run `\item`.

```

4886     \__enumext_start_list_tag:n {enumext*}
4887     \__enumext_fake_make_label_vii:n {#1}
4888     \__enumext_stop_start_list_tag:

```

Finally we open the `minipage` environment, capture the “*item content*”, make `\parindent` take the value of the key `listparindent` and `\parskip` take the value of the key `parsep`, then execute the keys `itemindent` and `first`.

Here the use of `\unskip` and `\skip_horizontal:n` with the value of `listparindent` is necessary, otherwise an unwanted space is created when using `\item[opt]` and the value passed to the key `itemindent` is incremented.

```

4889     \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
4890     \dim_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
4891     \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
4892     \__enumext_unskip_unkern:
4893     \__enumext_unskip_unkern:
4894     \skip_horizontal:n { -\l__enumext_listparindent_vii_dim } \ignorespaces
4895     \tl_use:N \l__enumext_fake_item_indent_vii_tl
4896     \tl_use:N \l__enumext_after_list_args_vii_tl
4897 }

```

The `__enumext_stop_item_vii:` function will finish the fetching `\item` and “*item content*” by closing the `minipage` environment, the *sockets* for *tagging* PDF and the *horizontal box*.

```

4898 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
4899 {
4900     \__enumext_endminipage:
4901     \__enumext_stop_list_tag:n {enumext*}
4902     \hbox_set_end:

```

Here we will reduce the *warnings* a bit by setting the value of `\hbadness` to `10000`, print `\item` and “*item content*” from the *horizontal box*.

```

4903     \int_set:Nn \hbadness { 10000 }
4904     \box_use_drop:N \l__enumext_item_text_vii_box

```

Finally apply the *vertical space* between rows set by `itemsep` key passed to `\parsep` using `\par\noindent` and *horizontal space* between columns set by `columns-sep` key using `\skip_horizontal:N`.

```

4905     \int_compare:nNnTF
4906     { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
4907     {
4908         \par\noindent
4909         \int_zero:N \l__enumext_item_column_pos_vii_int
4910     }
4911     {
4912         \skip_horizontal:N \l__enumext_columns_sep_vii_dim
4913     }
4914 }

```

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

`__enumext_remove_extra_parsep_vii:`

Remove the extra *vertical space* equal to `\parsep=\itemsep` when the total number of `\item` is divisible by the number of `\item` in the last row of the environment. Here the use of `\unskip` or `\removeatlastskip` fails and does not obtain the expected result, using `\vspace` is the option and in this case, we can use a simplified version since we are always in *(vertical mode)*.

```

4915 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
4916 {
4917     \int_compare:nNnT
4918     {
4919         \int_mod:nn
4920         { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
4921     }
4922     =
4923     { 0 }
4924     {
4925         \para_end:
4926         \skip_vertical:n { -\l__enumext_itemsep_vii_skip }
4927         \skip_vertical:N \c_zero_skip
4928         \int_gzero:N \g__enumext_item_count_all_vii_int
4929     }
4930 }

```

(End of definition for `__enumext_remove_extra_parsep_vii:.`)

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`.

```

4931 \__enumext_after_env:nn {enumext*}
4932 {
4933     \__enumext_execute_after_env:
4934 }

```

13.45 The environment `keyans*`

`keyans*`

The implementation of `keyans*` environment is the similar as that used by the `enumext*` environment except for the `__enumext_check_starred_cmd:n` function added in the second part.

```

4935 \NewDocumentEnvironment{keyans*}{ o }
4936 {
4937     \__enumext_safe_exec_viii:
4938     \__enumext_parse_keys_viii:n {#1}
4939     \__enumext_before_list_viii:
4940     \__enumext_start_list:nn { }
4941     {
4942         \__enumext_list_arg_two_viii:
4943         \__enumext_before_keys_exec_viii:
4944     }
4945     \IfDocumentMetadataTF { \tag_suspend:n {keyans*} } { } { }
4946     \__enumext_starred_columns_set_viii:
4947     \item[] \scan_stop:
4948     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_first_item_tmp_viii:
4949     \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
4950     \ignorespaces
4951 }
4952 {
4953     \IfDocumentMetadataTF { \tag_struct_end:n {tag=text-unit} } { } { }
4954     \__enumext_stop_item_tmp_viii:
4955     \__enumext_remove_extra_parsep_viii:
4956     \__enumext_check_starred_cmd:n { item }
4957     \__enumext_after_list_viii:
4958 }

```

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

`__enumext_safe_exec_viii:`

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

```

4959 \cs_new_protected:Nn \__enumext_safe_exec_viii:

```

```

4960 {
4961   \bool_if:NF \l__enumext_store_active_bool
4962   {
4963     \msg_error:nnnn { enumext } { wrong-place }{ keyans* }{ save-ans }
4964   }
4965   \int_incr:N \l__enumext_keyans_level_h_int
4966   \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
4967   {
4968     \msg_error:nn { enumext } { nested }
4969   }
4970   \__enumext_keyans_name_and_start:
4971   \bool_if:NT \l__enumext_starred_bool
4972   {
4973     \msg_error:nnn { enumext } { nested-horizontal } { enumext* }
4974   }
4975   \bool_set_true:N \l__enumext_starred_bool
4976   % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
4977   \bool_set_false:N \l__enumext_store_active_bool
4978   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
4979   {
4980     \msg_error:nn { enumext } { keyans-wrong-level }
4981   }
4982 }

```

(End of definition for `__enumext_safe_exec_viii:`)

```

\__enumext_parse_keys_viii:n Parse [<key = val>] for keyans*.
4983 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
4984 {
4985   \tl_if_novalue:nF {#1}
4986   {
4987     \keys_set:nn { enumext / keyans* } {#1}
4988   }
4989 }

```

(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`.

```

4990 \cs_new_protected:Nn \__enumext_before_list_viii:
4991 {
4992   \__enumext_vspace_above_viii:
4993   \__enumext_before_args_exec_viii:
4994   \__enumext_start_mini_viii:
4995 }

```

(End of definition for `__enumext_before_list_viii:`)

`__enumext_after_list_viii:` The function `__enumext_after_list_viii:` 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.

```

4996 \cs_new_protected:Nn \__enumext_after_list_viii:
4997 {
4998   \__enumext_stop_mini_viii:
4999   \__enumext_after_stop_list_viii:
5000   \__enumext_vspace_below_viii:
5001 }

```

(End of definition for `__enumext_after_list_viii:`)

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

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

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

```

5002 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_viii:
5003 {
5004     \skip_horizontal:n
5005     {
5006         -\__enumext_labelwidth_viii_dim - \__enumext_labelsep_viii_dim
5007     }
5008     \ignorespaces
5009 }

```

(End of definition for `__enumext_first_item_tmp_viii:`)

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

```

5010 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
5011 {
5012     \__enumext_stop_item_tmp_viii:
5013     \int_incr:N \l__enumext_item_column_pos_viii_int
5014     \int_gincr:N \g__enumext_item_count_all_viii_int
5015     \__enumext_item_peek_args_viii:
5016 }

```

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).

```

5017 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
5018 {
5019     \peek_meaning:NTF (
5020     { \__enumext_joined_item_viii:w }
5021     { \__enumext_joined_item_viii:w (1) }
5022 }

```

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`.

```

5023 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
5024 {
5025     \__enumext_starred_joined_item_viii:n {#1}
5026     \peek_meaning_remove:NTF *
5027     { \__enumext_starred_item_viii:w }
5028     { \__enumext_standar_item_viii:w }
5029 }

```

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]`.

```

5030 \cs_new_protected:Npn \__enumext_standar_item_viii:w
5031 {
5032     \bool_set_false:N \l__enumext_item_starred_viii_bool
5033     \bool_set_false:N \l__enumext_item_wrap_key_bool
5034     \peek_meaning:NTF [
5035     {
5036         \bool_set_eq:NN \l__enumext_wrap_label_viii_bool \l__enumext_wrap_label_opt_viii_bool
5037         \__enumext_start_item_viii:w
5038     }
5039     {
5040         \bool_set_true:N \l__enumext_wrap_label_viii_bool
5041         \legacy_if_set_true:n { @noitemarg }
5042         \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ] \ignorespaces
5043     }
5044 }

```

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

```
\__enumext_starred_item_viii:w
\__enumext_starred_item_viii_aux_i:w
\__enumext_starred_item_viii_aux_ii:w
\__enumext_keyans_starred_item_star:
```

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

```
5045 \cs_new_protected:Npn \__enumext_starred_item_viii:w
5046 {
5047   \bool_set_true:N \l__enumext_item_starred_viii_bool
5048   \bool_set_true:N \l__enumext_item_wrap_key_bool
5049   \bool_set_true:N \l__enumext_wrap_label_viii_bool
5050   \peek_meaning:NTF [
5051     { \__enumext_starred_item_viii_aux_i:w }
5052     { \__enumext_starred_item_viii_aux_ii:w }
5053   }
```

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`.

```
5054 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
5055 {
5056   \tl_clear:N \l__enumext_store_current_label_tl
5057   \tl_if_no_value:nF { #1 }
5058   {
5059     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
5060     {
5061       \tl_put_right:Ne \l__enumext_store_current_label_tl
5062       {
5063         \l__enumext_store_keyans_item_opt_sep_tl
5064       }
5065       \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
5066     }
5067     \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
5068   }
5069   \__enumext_starred_item_viii_aux_ii:w
5070 }
5071 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
5072 {
5073   \legacy_if_set_true:n { @noitemarg }
5074   \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ] \ignorespaces
5075 }
```

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

```
5076 \cs_new_protected:Nn \__enumext_keyans_starred_item_star:
5077 {
5078   \tl_put_left:Ne \l__enumext_store_current_label_tl { \l__enumext_label_viii_tl }
5079   \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
5080   \__enumext_keyans_store_ref:
5081   \tl_put_left:Ne \l__enumext_store_current_label_tl { \item }
5082   \__enumext_keyans_addto_seq_link:
5083   \int_gincr:N \g__enumext_check_starred_cmd_int
5084   \dim_compare:nNnT { \l__enumext_mark_sym_sep_dim } = { \c_zero_dim }
5085   {
5086     \dim_set:Nn \l__enumext_mark_sym_sep_dim { \l__enumext_labelsep_viii_dim }
5087   }
5088   \bool_if:NT \l__enumext_show_answer_bool
5089   {
5090     \__enumext_print_keyans_box:NN
5091     \l__enumext_labelwidth_viii_dim \l__enumext_mark_sym_sep_dim
5092   }
5093   \bool_if:NT \l__enumext_show_position_bool
5094   {
5095     \tl_set:Ne \l__enumext_mark_answer_sym_tl
5096     {
5097       \group_begin:
5098       \exp_not:N \normalfont
5099       \exp_not:N \footnotesize [ \int_eval:n
5100       {
5101         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }

```

```

5102         }
5103     ]
5104     \group_end:
5105 }
5106 \__enumext_print_keyans_box:NN
5107 \l__enumext_labelwidth_viii_dim \l__enumext_mark_sym_sep_dim
5108 }
5109 }

```

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

```

\__enumext_keyans_wrapper_label_viii:n
\__enumext_fake_make_label_viii:n

```

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

```

5110 \cs_new_protected:Npn \__enumext_keyans_wrapper_label_viii:n #1
5111 {
5112     \bool_lazy_all:nT
5113     {
5114         { \bool_if_p:N \l__enumext_wrap_label_viii_bool }
5115         { \bool_if_p:N \l__enumext_show_answer_bool }
5116         { \bool_if_p:N \l__enumext_item_wrap_key_bool }
5117         { \cs_if_exist_p:N \__enumext_keyans_wrapper_item:n }
5118     }
5119     {
5120         \cs_set_eq:NN \__enumext_wrapper_label_viii:n \__enumext_keyans_wrapper_item:n
5121     }
5122     \bool_if:NTF \l__enumext_wrap_label_viii_bool
5123     {
5124         \__enumext_wrapper_label_viii:n {#1}
5125     }
5126     { #1 }
5127 }
5128 \cs_new_protected_nopar:Npn \__enumext_fake_make_label_viii:n #1
5129 {
5130     \legacy_if:nT { @noitemarg }
5131     {
5132         \legacy_if_set_false:n { @noitemarg }
5133         \legacy_if:nT { @nmbrlist }
5134         {
5135             \refstepcounter{enumXviii}
5136         }
5137     }
5138     \bool_if:NT \l__enumext_item_starred_viii_bool
5139     {
5140         \__enumext_keyans_starred_item_star:
5141     }
5142     \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
5143     {
5144         \tl_use:N \l__enumext_label_font_style_viii_tl
5145         \__enumext_keyans_wrapper_label_viii:n {#1}
5146     }
5147     \skip_horizontal:N \l__enumext_labelsep_viii_dim \ignorespaces
5148 }

```

(End of definition for `__enumext_keyans_wrapper_label_viii:n` and `__enumext_fake_make_label_viii:n`.)

13.45.2 Real definition of `\item` in `keyans*`

```

\__enumext_start_item_viii:w
\__enumext_stop_item_viii:

```

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

```

5149 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
5150 {
5151     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
5152     \hbox_set_to_wd:Nnw \l__enumext_item_text_viii_box
5153     {
5154         \l__enumext_joined_width_viii_dim
5155         + \l__enumext_labelwidth_viii_dim
5156         + \l__enumext_labelsep_viii_dim
5157     }
5158     \__enumext_renew_footnote_starred:
5159     \__enumext_start_list_tag:n {keyans*}
5160     \__enumext_fake_make_label_viii:n {#1}
5161     \__enumext_stop_start_list_tag:
5162     \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
5163     \dim_set_eq:NN \parindent \l__enumext_listparindent_viii_dim

```

```

5164     \skip_set_eq:Nn \parskip \l__enumext_parsep_viii_skip
5165     \__enumext_unskip_unkern:
5166     \__enumext_unskip_unkern:
5167     \skip_horizontal:n { -\l__enumext_listparindent_viii_dim } \ignorespaces
5168     \tl_use:N \l__enumext_fake_item_indent_viii_tl
5169     \bool_if:NT \l__enumext_item_starred_viii_bool
5170     {
5171         \__enumext_keyans_show_item_opt:
5172     }
5173     \tl_use:N \l__enumext_after_list_args_viii_tl
5174 }
5175 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
5176 {
5177     \__enumext_endminipage:
5178     \__enumext_stop_list_tag:n {keyans*}
5179     \hbox_set_end:
5180     \int_set:Nn \hbadness { 10000 }
5181     \box_use_drop:N \l__enumext_item_text_viii_box
5182     \int_compare:nNnTF
5183     { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
5184     {
5185         \par\noindent
5186         \int_zero:N \l__enumext_item_column_pos_viii_int
5187     }
5188     {
5189         \skip_horizontal:N \l__enumext_columns_sep_viii_dim
5190     }
5191 }

```

(End of definition for __enumext_start_item_viii:w and __enumext_stop_item_viii:.)

__enumext_remove_extra_parsep_viii:

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

```

5192 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
5193 {
5194     \int_compare:nNnT
5195     {
5196         \int_mod:nn
5197         { \g__enumext_item_count_all_viii_int }
5198         { \l__enumext_columns_viii_int }
5199     }
5200     =
5201     { 0 }
5202     {
5203         \para_end:
5204         \skip_vertical:n { -\l__enumext_itemsep_viii_skip }
5205         \skip_vertical:N \c_zero_skip
5206         \int_gzero:N \g__enumext_item_count_all_viii_int
5207     }
5208 }

```

(End of definition for __enumext_remove_extra_parsep_viii:.)

13.46 The command \getkeyans

\getkeyans

__enumext_getkeyans_aux:n

__enumext_getkeyans:n

The `\getkeyans` command takes a *mandatory argument* of the form $\langle \textit{store name} : \textit{position} \rangle$. Retrieve a “single content” stored by `\anskey`, `\anspic*` and `\item*` and `anskey*` from *prop list* defined by `save-anskey`.

```

5209 \NewDocumentCommand \getkeyans { m }
5210 {
5211     \exp_args:Ne \__enumext_getkeyans_aux:n
5212     { \tl_to_str:e { \text_expand:n {#1} } }
5213 }

```

The internal function `__enumext_getkeyans_aux:n` is in charge of *splitting* the *mandatory argument* using “:”. If “:” is omitted it will return an error.

```

5214 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
5215 {
5216     \str_if_in:nnTF {#1} { : }
5217     {
5218         \use:e
5219         {
5220             \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:

```

```

5221         { {##1} {##2} }
5222     }
5223     \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
5224 }
5225 { \msg_error:nnn { enumext } { missing-colon } {#1} }
5226 }

```

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

```

5227 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
5228 {
5229     \prop_if_exist:cTF { g__enumext_#1_prop }
5230     {
5231         \prop_item:cn { g__enumext_#1_prop }{#2}
5232     }
5233     {
5234         \msg_error:nnn { enumext } { undefined-storage-anskey } {#1}
5235     }
5236 }

```

(End of definition for `\getkeyans`, `__enumext_getkeyans_aux:n`, and `__enumext_getkeyans:nn`. This function is documented on page 18.)

13.47 The command `\printkeyans`

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

```

5237 \keys_define:nn { enumext / print }
5238 {
5239     print* .code:n      = \keys_precompile:neN { enumext / enumext* }
5240               { \__enumext_filter_save_key:n {#1} }
5241               \l__enumext_print_keyans_starred_tl, % starred cmd
5242     print* .initial:n   = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
5243     print-1 .code:n     = \keys_precompile:neN { enumext / level-1 }
5244               { \__enumext_filter_save_key:n {#1} }
5245               \l__enumext_print_keyans_i_tl,
5246     print-1 .initial:n  = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
5247     print-2 .code:n     = \keys_precompile:neN { enumext / level-2 }
5248               { \__enumext_filter_save_key:n {#1} }
5249               \l__enumext_print_keyans_ii_tl,
5250     print-2 .initial:n  = { nosep, label=(\alph*), first=\small, font=\small },
5251     print-3 .code:n     = \keys_precompile:neN { enumext / level-3 }
5252               { \__enumext_filter_save_key:n {#1} }
5253               \l__enumext_print_keyans_iii_tl,
5254     print-3 .initial:n  = { nosep, label=\roman*., first=\small, font=\small },
5255     print-4 .code:n     = \keys_precompile:neN { enumext / level-4 }
5256               { \__enumext_filter_save_key:n {#1} }
5257               \l__enumext_print_keyans_iv_tl,
5258     print-4 .initial:n  = { nosep, label=\Alph*., first=\small, font=\small },
5259     print-* .code:n     = \keys_precompile:neN { enumext / enumext* }
5260               { \__enumext_filter_save_key:n {#1} }
5261               \l__enumext_print_keyans_vii_tl, % starred nested
5262     print-* .initial:n  = { nosep, label=\arabic*., first=\small, font=\small },
5263 }

```

- 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
\__enumext_printkeyans:nnn

```

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`.

```

5264 \NewDocumentCommand \printkeyans { s O{} m }

```

```

5265 {
5266   \group_begin:
5267     \tl_use:N \l__enumext_print_keyans_i_tl
5268     \tl_use:N \l__enumext_print_keyans_ii_tl
5269     \tl_use:N \l__enumext_print_keyans_iii_tl
5270     \tl_use:N \l__enumext_print_keyans_iv_tl
5271     \tl_use:N \l__enumext_print_keyans_vii_tl
5272     \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
5273   \group_end:
5274 }

```

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.

```

5275 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
5276 {
5277   \seq_if_exist:cTF { g__enumext_#3_seq }
5278   {
5279     \seq_if_empty:cF { g__enumext_#3_seq }
5280     {

```

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*`, we set `\l__enumext_base_line_fix_bool` and `\l__enumext_print_keyans_star_bool` to true for *baseline correction*, open the `enumext*` environment passing the *optional argument* and map the *sequence*, then set `\l__enumext_base_line_fix_bool` and `\l__enumext_print_keyans_star_bool` to false.

```

5281     \bool_if:nTF {#1}
5282     {
5283       \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
5284       {
5285         \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
5286       }
5287       {
5288         \tl_use:N \l__enumext_print_keyans_starred_tl
5289         \bool_set_true:N \l__enumext_base_line_fix_bool
5290         \bool_set_true:N \l__enumext_print_keyans_star_bool
5291         \begin{enumext*}[#2]
5292           \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5293           \end{enumext*}
5294           \bool_set_false:N \l__enumext_base_line_fix_bool
5295           \bool_set_false:N \l__enumext_print_keyans_star_bool
5296         }
5297       }

```

Otherwise it will open the environment `enumext` passing the *optional argument* to the “first level” then map the *sequence*.

```

5298       {
5299         \begin{enumext}[#2]
5300         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5301         \end{enumext}
5302       }
5303     }
5304   }
5305   {
5306     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
5307   }
5308 }

```

(End of definition for `\printkeyans` and `__enumext_printkeyans:nnn`. This function is documented on page 18.)

13.48 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`.

```

5309 \cs_new:Npn \__enumext_filter_first_level:n #1
5310 {
5311   \use:e
5312   {

```

```

5313         \keyval_parse:NNn
5314         __enumext_filter_first_level_key:n
5315         __enumext_filter_first_level_pair:nn {#1}
5316     }
5317 }

```

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*`.

```

5318 \cs_new:Npn __enumext_filter_first_level_key:n #1
5319 {
5320     \str_case:nnF {#1}
5321     {
5322         { resume } {}
5323         { resume* } {}
5324     }
5325     { , { \exp_not:n {#1} } }
5326 }

```

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.

```

5327 \cs_new:Npn __enumext_filter_first_level_pair:nn #1#2
5328 {
5329     \str_case:nnF {#1}
5330     {
5331         { series } {}
5332         { resume } {}
5333         { save-ans } {}
5334     }
5335     { , { \exp_not:n {#1} } } = { \exp_not:n {#2} } }
5336 }

```

(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`.

```

5337 \keys_define:nn { enumext / meta-families }
5338 {
5339     enumext-1 .code:n =
5340     {
5341         \keys_set:ne { enumext / level-1 }
5342         {
5343             __enumext_filter_first_level:n {#1}
5344         }
5345     } ,
5346     enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
5347     enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
5348     enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
5349     keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} } ,
5350     enumext* .code:n =
5351     {
5352         \keys_set:ne { enumext / enumext* }
5353         {
5354             __enumext_filter_first_level:n {#1}
5355         }
5356     } ,
5357     keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
5358     print* .code:n = { \keys_set:nn { enumext / print } { print* = {#1} } } ,
5359     print-1 .code:n = { \keys_set:nn { enumext / print } { print-1 = {#1} } } ,
5360     print-2 .code:n = { \keys_set:nn { enumext / print } { print-2 = {#1} } } ,
5361     print-3 .code:n = { \keys_set:nn { enumext / print } { print-3 = {#1} } } ,
5362     print-4 .code:n = { \keys_set:nn { enumext / print } { print-4 = {#1} } } ,
5363     print-* .code:n = { \keys_set:nn { enumext / print } { print-* = {#1} } } ,
5364     unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
5365 }

```

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

```

5366 \seq_const_from_clist:Nn \c__enumext_all_families_seq
5367 {
5368     enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
5369     keyans*, print-1, print-2, print-3, print-4, print-*, print*,
5370 }

```



```

\setenumext
__enumext_set_parse:n
__enumext_set_error:nn

5371 \NewDocumentCommand \setenumext { 0{enumext,1} +m }
5372 {
5373   \seq_clear:N \__enumext_setkey_tmpa_seq
5374   \seq_set_from_clist:Nn \__enumext_setkey_tmpb_seq {#1}
5375   \int_set:Nn \__enumext_setkey_tmpa_int
5376   {
5377     \seq_count:N \__enumext_setkey_tmpb_seq
5378   }
5379   \int_compare:nNnTF { \__enumext_setkey_tmpa_int } > { 1 }
5380   {
5381     \seq_pop_left:NN \__enumext_setkey_tmpb_seq \__enumext_setkey_tmpa_tl
5382     \seq_map_function:NN \__enumext_setkey_tmpb_seq \__enumext_set_parse:n
5383     \seq_set_map_e:NNn \__enumext_setkey_tmpa_seq \__enumext_setkey_tmpa_seq
5384     {
5385       \tl_use:N \__enumext_setkey_tmpa_tl - ##1
5386     }
5387   }
5388   {
5389     \seq_put_right:Ne \__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
5390   }
5391   \seq_if_empty:NNTF \__enumext_setkey_tmpa_seq
5392   { \seq_map_inline:Nn \c__enumext_all_families_seq }
5393   { \seq_map_inline:Nn \__enumext_setkey_tmpa_seq }
5394   {
5395     \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
5396   }
5397 }

```

Internal functions used by the `\setenumext` command.

```

5398 \cs_new_protected:Npn \__enumext_set_parse:n #1
5399 {
5400   \tl_set:Ne \__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
5401   \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
5402   { \tl_remove_all:Nn \__enumext_setkey_tmpb_tl {##1} }
5403   \tl_if_empty:NNTF \__enumext_setkey_tmpb_tl
5404   {
5405     \seq_put_right:Ne \__enumext_setkey_tmpa_seq
5406     { \tl_trim_spaces:n {#1} }
5407   }
5408   { \__enumext_set_error:nn {#1} { } }
5409 }
5410 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
5411 { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }

```

(End of definition for `\setenumext`, `__enumext_set_parse:n`, and `__enumext_set_error:nn`. This function is documented on page 6.)

13.49 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
\c__enumext_meta_paths_prop
__enumext_add_meta_key:nnn
__enumext_def_meta_key:nnn
__enumext_def_meta_key:Vnn

5412 \prop_const_from_keyval:Nn \c__enumext_meta_paths_prop
5413 {
5414   {enumext,1} = level-1,
5415   {enumext,2} = level-2,
5416   {enumext,3} = level-3,
5417   {enumext,4} = level-4,
5418   {enumext*} = enumext*
5419 }

```

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

```

5420 \NewDocumentCommand \setenumextmeta { s 0{enumext,1} m +m }
5421 {
5422   \str_if_eq:eeTF { \tl_trim_spaces:n {#3} } { unknown }
5423   { \msg_error:nn { enumext } { prohibited-unknown } }
5424   {
5425     \bool_if:nTF {#1}
5426     {
5427       \int_step_inline:nn { 4 }

```

```

5428         { \__enumext_add_meta_key:nnn { enumext, ##1 } {#3} {#4} }
5429         \__enumext_add_meta_key:nnn { enumext* } {#3} {#4}
5430     }
5431     { \__enumext_add_meta_key:nnn {#2} {#3} {#4} }
5432 }
5433 }

```

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

```

5434 \cs_new_protected:Npn \__enumext_add_meta_key:nnn #1
5435 {
5436     \tl_set:Nn \l__enumext_meta_path_tl {#1}
5437     \tl_replace_all:Nnn \l__enumext_meta_path_tl { ~ } {}
5438     \prop_get:NVNTF
5439     \c__enumext_meta_paths_prop \l__enumext_meta_path_tl \l__enumext_meta_path_tl
5440     { \__enumext_def_meta_key:Vnn \l__enumext_meta_path_tl }
5441     {
5442         \msg_error:nnn { enumext } { unknown-set } {#1}
5443         \use_none:nn
5444     }
5445 }
5446 \cs_new_protected:Npn \__enumext_def_meta_key:nnn #1#2#3
5447 {
5448     \bool_lazy_or:nnTF
5449     { \keys_if_exist:p:nn { enumext / #1 } {#2} }
5450     { \keys_if_exist:p:nn { enumext / enumext* } {#2} }
5451     { \msg_error:nnn { enumext } { already-defined } {#2} }
5452     {
5453         \keys_define:nn { enumext / #1 }
5454         {
5455             #2 .meta:n = {#3},
5456             #2 .value_forbidden:n = true
5457         }
5458     }
5459 }
5460 \cs_generate_variant:Nn \__enumext_def_meta_key:nnn { V }

```

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

13.50 The command `\foreachkeyans`

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

`\foreachkeyans`

```

\__enumext_parse_foreach_keys:nn
\__enumext_parse_foreach_keys:n
\__enumext_foreach_keyans:nn
\__enumext_foreach_add_body:n

```

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

```

5461 \keys_define:nn { enumext / foreach }
5462 {
5463     before .tl_set:N = \l__enumext_foreach_before_tl,
5464     before .value_required:n = true,
5465     after .tl_set:N = \l__enumext_foreach_after_tl,
5466     after .value_required:n = true,
5467     start .int_set:N = \l__enumext_foreach_start_int,
5468     start .value_required:n = true,
5469     stop .int_set:N = \l__enumext_foreach_stop_int,
5470     stop .value_required:n = true,
5471     step .int_set:N = \l__enumext_foreach_step_int,
5472     step .value_required:n = true,
5473     wrapper .cs_set_protected:Np = \__enumext_foreach_wrapper:n #1,
5474     wrapper .value_required:n = true,
5475     sep .tl_set:N = \l__enumext_foreach_sep_tl,
5476     sep .value_required:n = true,
5477     unknown .code:n = { \__enumext_parse_foreach_keys:n {#1} }
5478 }
5479 \keys_precompile:nnN { enumext / foreach }
5480 {
5481     before={},after={},start=1,step=1,stop=0,wrapper=#1,sep={} }
5482 }
5483 \g__enumext_foreach_default_keys_tl

```

Functions for handling unknown $\langle keys \rangle$.

```

5484 \cs_new_protected:Npn \__enumext_parse_foreach_keys:nn #1#2
5485 {
5486   \tl_if_blank:nTF {#2}
5487   {
5488     \msg_error:nnn { enumext } { for-key-unknown } {#1}
5489   }
5490   {
5491     \msg_error:nnnn { enumext } { for-key-value-unknown } {#1} {#2}
5492   }
5493 }
5494 \cs_new_protected:Npn \__enumext_parse_foreach_keys:n #1
5495 {
5496   \exp_args:NV \__enumext_parse_foreach_keys:nn \l_keys_key_str {#1}
5497 }

```

We create the command.

```

5498 \NewDocumentCommand \foreachkeyans { +0{} m }
5499 {
5500   \__enumext_foreach_keyans:nn {#1} {#2}
5501 }

```

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

```

5502 \cs_new_protected:Npn \__enumext_foreach_keyans:nn #1 #2
5503 {
5504   \tl_use:N \g__enumext_foreach_default_keys_tl
5505   \keys_set:nn { enumext / foreach } {#1}
5506   \tl_set:Nn \l__enumext_foreach_name_prop_tl {#2}
5507   \prop_if_exist:cF { g__enumext_#2_prop }
5508   {
5509     \msg_error:nnn { enumext } { undefined-storage-anskey } {#2}
5510   }
5511   \int_compare:nNt { \l__enumext_foreach_stop_int } = { 0 }
5512   {
5513     \int_set:Nn \l__enumext_foreach_stop_int
5514     { \prop_count:c { g__enumext_#2_prop } }
5515   }
5516   \seq_clear:N \l__enumext_foreach_print_seq
5517   \int_step_function:nnnN
5518   { \l__enumext_foreach_start_int }
5519   { \l__enumext_foreach_step_int }
5520   { \l__enumext_foreach_stop_int }
5521   \__enumext_foreach_add_body:n
5522   \seq_use:NV \l__enumext_foreach_print_seq \l__enumext_foreach_sep_tl
5523 }
5524 \cs_new_protected:Npn \__enumext_foreach_add_body:n #1
5525 {
5526   \seq_put_right:Ne \l__enumext_foreach_print_seq
5527   {
5528     \exp_not:V \l__enumext_foreach_before_tl
5529     \__enumext_foreach_wrapper:n
5530     {
5531       \prop_item:cn { g__enumext_ \l__enumext_foreach_name_prop_tl _prop }{#1}
5532     }
5533     \exp_not:V \l__enumext_foreach_after_tl
5534   }
5535 }

```

(End of definition for `\foreachkeyans` and others. This function is documented on page 18.)

13.51 Messages

Message used by package-load for `multicol` and `hyperref` packages.

```

5536 \msg_new:nnn { enumext } { package-load }
5537 {
5538   The ~ '#1' ~ package ~ is ~ already ~ loaded.
5539 }
5540 \msg_new:nnn { enumext } { package-not-load }
5541 {
5542   The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
5543 }

```

```

5544 \msg_new:nnn { enumext } { package-load-foot }
5545 {
5546   The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
5547 }

```

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

```

5548 \msg_new:nnn { enumext } { counters }
5549 {
5550   The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
5551   package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
5552 }

```

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

```

5553 \msg_new:nnn { enumext } { unknown-choice }
5554 {
5555   The ~ value ~ '#3' ~ for ~ '#1' ~ key ~ is ~ invalid ~ use ~ ('#2').
5556 }

```

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

```

5557 \msg_new:nnnn { enumext } { anskey-env-error }
5558 {
5559   The ~ '#1' ~ environment ~is ~ reserved ~ by ~\\
5560   'enumext' ~ package, ~ It~ is~ already~ defined.
5561 }
5562 {
5563   The ~ anskey* ~ environment ~ is ~ defined ~ internally ~
5564   for ~ the ~ 'save-ans' ~ key.\\
5565 }

```

Message used in the creation of *prop list* by **enumext** package.

```

5566 \msg_new:nnn { enumext } { store-prop }
5567 {
5568   * ~ Package ~ enumext: ~ Creating ~
5569   \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
5570 }
5571 \msg_new:nnn { enumext } { store-seq }
5572 {
5573   * ~ Package ~ enumext: ~ Creating ~
5574   \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
5575 }
5576 \msg_new:nnn { enumext } { store-int }
5577 {
5578   * ~ Package ~ enumext: ~ Creating ~
5579   \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_context:.
5580 }
5581 \msg_new:nnn { enumext } { prop-seq-int-hook }
5582 {
5583   * ~ Package ~ enumext: ~ Elements ~ in ~
5584   \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
5585   * ~ Package ~ enumext: ~ Elements ~ in ~
5586   \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
5587   * ~ Package ~ enumext: ~ Value ~ off ~
5588   \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
5589 }
5590 \msg_new:nnn { enumext } { item-answer-hook }
5591 {
5592   * ~ Package ~ enumext: ~ Value ~ off ~
5593   \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\\
5594   * ~ Package ~ enumext: ~ Value ~ off ~
5595   \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\\
5596   * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
5597 }

```

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

```

5598 \msg_new:nnn { enumext } { invalid-key }
5599 {
5600   The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
5601 }
5602 \msg_new:nnn { enumext } { unknown-key-family }
5603 {
5604   Unknown~key~family~`\l_keys_key_str'~for~enumext.
5605 }

```

Messages used in length calculation.

```

5606 \msg_new:nnn { enumext } { width-negative }
5607 {
5608   Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
5609   The ~ key ~ '#1'~ accepts ~ values ~ >= ~ opt.
5610 }
5611 \msg_new:nnn { enumext } { width-zero }
5612 {
5613   Invalid ~ '#1=#2' ~ \msg_line_context:.\
5614   The ~ key ~ '#1'~ accepts ~ values ~ > ~ opt.
5615 }

```

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

```

5616 \msg_new:nnn { enumext } { list-lengths }
5617 {
5618   **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\
5619   \__enumext_show_length:nnn { dim } { labelsep } {#1}
5620   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
5621   \__enumext_show_length:nnn { dim } { itemindent } {#1}
5622   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
5623   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
5624   \__enumext_show_length:nnn { dim } { listparindent } {#1}
5625   \__enumext_show_length:nnn { skip } { topsep } {#1}
5626   \__enumext_show_length:nnn { skip } { parsep } {#1}
5627   \__enumext_show_length:nnn { skip } { partopsep } {#1}
5628   \__enumext_show_length:nnn { skip } { itemsep } {#1}
5629   ****
5630 }

```

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

```

5631 \msg_new:nnn { enumext } { list-lengths-not-nested }
5632 {
5633   **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\
5634   \__enumext_show_length:nnn { dim } { labelsep } {#1}
5635   \__enumext_show_length:nnn { dim } { labelwidth } {#1}
5636   \__enumext_show_length:nnn { dim } { itemindent } {#1}
5637   \__enumext_show_length:nnn { dim } { leftmargin } {#1}
5638   \__enumext_show_length:nnn { dim } { rightmargin } {#1}
5639   \__enumext_show_length:nnn { dim } { listparindent } {#1}
5640   \__enumext_show_length:nnn { skip } { topsep } {#1}
5641   \__enumext_show_length:nnn { skip } { parsep } {#1}
5642   \__enumext_show_length:nnn { skip } { partopsep } {#1}
5643   \__enumext_show_length:nnn { skip } { itemsep } {#1}
5644   ****
5645 }

```

Messages used by `ref` key.

```

5646 \msg_new:nnn { enumext } { key-ref-empty }
5647 {
5648   Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
5649 }

```

Messages used by `save-ans` key.

```

5650 \msg_new:nnn { enumext } { save-ans-empty }
5651 {
5652   Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
5653 }
5654 \msg_new:nnn { enumext } { save-ans-log }
5655 {
5656   * ~ Package ~ enumext: ~ Start ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5657 }
5658 \msg_new:nnn { enumext } { save-ans-log-hook }
5659 {
5660   * ~ Package ~ enumext: ~ Stop ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5661 }
5662 \msg_new:nnn { enumext } { save-ans-hook }
5663 {
5664   Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
5665 }

```

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

```

5666 \msg_new:nnn { enumext } { need-save-ans }
5667 {

```

```

5668     Key ~ '#1' ~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2' ~ \msg_line_context:.
5669 }
5670 \msg_new:nnn { enumext } { items-same-answer }
5671 {
5672     *****\
5673     * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~
5674     for ~ \c_left_brace_str #2 \c_right_brace_str\
5675     * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~
5676     'OK', ~ all ~ items ~ with ~ answer.\
5677     *****
5678 }
5679 \msg_new:nnn { enumext } { item-greater-answer }
5680 {
5681     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\
5682     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~ 'NOT ~ OK'\
5683     Items ~ > ~ Answers.
5684 }
5685 \msg_new:nnn { enumext } { item-less-answer }
5686 {
5687     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\
5688     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~ 'NOT ~ OK'\
5689     Items ~ < ~ Answers.
5690 }

```

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

```

5691 \msg_new:nnn { enumext } { missing-starred }
5692 {
5693     Missing ~ '\c_backslash_str #1*' ~ #2.
5694 }
5695 \msg_new:nnn { enumext } { many-starred }
5696 {
5697     Many ~ '\c_backslash_str #1*' ~ #2.
5698 }

```

Messages used by `\printkeyans*` command.

```

5699 \msg_new:nnn { enumext } { print-starred }
5700 {
5701     \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
5702     #2 ~ environment ~ \msg_line_context:.
5703 }

```

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

```

5704 \msg_new:nnn { enumext } { list-too-deep }
5705 {
5706     Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \
5707     The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
5708 }

```

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

```

5709 \msg_new:nnn { enumext } { anskey-unnumber-item }
5710 {
5711     Can't ~ store ~ with ~ a ~ unnumbered ~ \c_backslash_str item ~ \msg_line_context:.
5712 }
5713 \msg_new:nnn { enumext } { anskey-already-stored }
5714 {
5715     Content ~ already ~ stored ~ for ~ this ~ \c_backslash_str item ~ \msg_line_context:.
5716 }
5717 \msg_new:nnn { enumext } { anskey-empty-arg }
5718 {
5719     Can't ~ store ~ empty ~ content ~ \msg_line_context:.
5720 }
5721 \msg_new:nnn { enumext } { anskey-wrong-place }
5722 {
5723     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \
5724     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5725 }
5726 \msg_new:nnn { enumext } { anskey-nested }
5727 {
5728     The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
5729 }
5730 \msg_new:nnn { enumext } { anskey-math-mode }
5731 {
5732     #1 ~ can't ~ work ~ in ~ math ~ mode ~ \msg_line_context:.

```

```

5733     }
5734     \msg_new:nnn { enumext } { anskey-env-wrong }
5735     {
5736         The ~ environment ~ anskey* ~ cannot ~ use ~ in ~ '#1' ~ \msg_line_context:.
5737     }
5738     \msg_new:nnn { enumext } { ansPIC-wrong-place }
5739     {
5740         Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5741         '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5742     }
5743     \msg_new:nnn { enumext } { command-wrong-place }
5744     {
5745         Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5746         '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
5747     }
5748     \msg_new:nnnn { enumext } { anskey-env-key-unknown }
5749     {
5750         The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5751         'anskey*' ~ and ~ is ~ being ~ ignored.
5752     }
5753     {
5754         The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5755         Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5756     }
5757     \msg_new:nnnn { enumext } { anskey-env-key-value-unknown }
5758     {
5759         The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5760         'anskey*' ~ and ~ is ~ being ~ ignored.
5761     }
5762     {
5763         The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5764         Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5765     }
5766     \msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
5767     { The ~ key ~ '#1' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.}
5768     {
5769         The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5770         Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5771     }
5772     \msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
5773     { The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.}
5774     {
5775         The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5776         Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5777     }

```

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

```

5778     \msg_new:nnn { enumext } { keyans-nested }
5779     {
5780         The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
5781     }
5782     \msg_new:nnn { enumext } { keyans-wrong-level }
5783     {
5784         Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
5785         The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5786     }
5787     \msg_new:nnn { enumext } { wrong-place }
5788     {
5789         Wrong ~ place ~ for ~ '#1' ~ environment ~\msg_line_context:~ \\
5790         '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext.
5791     }
5792     \msg_new:nnn { enumext } { keyansPIC-nested }
5793     {
5794         The ~ environment ~ 'keyansPIC' ~ can't ~ be ~ nested~ \msg_line_context:~.
5795     }
5796     \msg_new:nnn { enumext } { keyansPIC-wrong-level }
5797     {
5798         Wrong ~ level ~ position ~ for ~ 'keyansPIC' ~ \msg_line_context:~ \\
5799         The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5800     }
5801     \msg_new:nnn { enumext } { keyansPIC-item-cmd }
5802     {

```



```

5803     Can't ~ use ~ \c_backslash_str item ~ in ~ keyanspic ~ \msg_line_context:.
5804 }
5805 \msg_new:nnnn { enumext } { keyans-unknown-key }
5806 {
5807     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5808     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5809 }
5810 {
5811     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5812     ~ have ~ a ~ key ~ called ~'#1'.\\
5813     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5814 }
5815 \msg_new:nnnn { enumext } { keyans-unknown-key-value }
5816 {
5817     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5818     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5819 }
5820 {
5821     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5822     ~ have ~ a ~ key ~ called ~'#1'.\\
5823     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5824 }

```

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

```

5825 \msg_new:nnnn { enumext } { starred-unknown-key }
5826 {
5827     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5828     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5829 }
5830 {
5831     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5832     ~ have ~ a ~ key ~ called ~'#1'.\\
5833     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5834 }
5835 \msg_new:nnnn { enumext } { starred-unknown-key-value }
5836 {
5837     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5838     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5839 }
5840 {
5841     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5842     ~ have ~ a ~ key ~ called ~'#1'.\\
5843     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5844 }

```

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

```

5845 \msg_new:nnnn { enumext } { standar-unknown-key }
5846 {
5847     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_tl
5848     ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5849 }
5850 {
5851     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5852     ~ have ~ a ~ key ~ called ~'#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5853     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5854 }
5855 \msg_new:nnnn { enumext } { standar-unknown-key-value }
5856 {
5857     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_
5858     ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5859 }
5860 {
5861     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5862     ~ have ~ a ~ key ~ called ~'#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5863     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5864 }

```

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

```

5865 \msg_new:nnnn { enumext } { for-key-unknown }
5866 { The~key~'#1'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored.}
5867 {
5868     The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\

```

```

5869     Check~that~you~have~spelled~the~key~name~correctly.
5870 }
5871 \msg_new:nnnn { enumext } { for-key-value-unknown }
5872 { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored. }
5873 {
5874     The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5875     Check~that~you~have~spelled~the~key~name~correctly.
5876 }

```

Messages used by `\getkeyans` command.

```

5877 \msg_new:nnn { enumext } { undefined-storage-anskey }
5878 {
5879     Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
5880 }

```

Messages used by `\miniright` command.

```

5881 \msg_new:nnn { enumext } { missing-miniright }
5882 {
5883     Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\\
5884     The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
5885 }
5886 \msg_new:nnn { enumext } { wrong-miniright-place }
5887 {
5888     Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
5889     Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
5890 }
5891 \msg_new:nnn { enumext } { wrong-miniright-use }
5892 {
5893     Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \\
5894     '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
5895 }
5896 \msg_new:nnn { enumext } { wrong-miniright-starred }
5897 {
5898     Can't ~ use ~ \c_backslash_str miniright ~ in ~ starred ~ environments ~ \msg_line_context:.
5899 }
5900 \msg_new:nnn { enumext } { many-miniright-used }
5901 {
5902     Can't ~ use ~ \c_backslash_str miniright ~ more ~ than ~ once ~ \msg_line_context:.
5903 }

```

Messages used by `\setenumextmeta` command.

```

5904 \msg_new:nnn { enumext } { unknown-set }
5905 {
5906     Argument ~ [#1] ~ is ~ unknown ~ by ~ \c_backslash_str setenumextmeta ~ \msg_line_context:.
5907 }
5908 \msg_new:nnn { enumext } { already-defined }
5909 {
5910     The ~ key ~ '#1' ~ is ~ already ~ defined ~ \msg_line_context:.
5911 }
5912 \msg_new:nnn { enumext } { prohibited-unknown }
5913 {
5914     The ~ name ~ 'unknown' ~ can't ~ be ~ chosen~ for ~ a ~ meta ~ key ~ \msg_line_context:.
5915 }

```

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

```

5916 \msg_new:nnn { enumext } { nested }
5917 {
5918     The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ \msg_line_con
5919 }
5920 \msg_new:nnn { enumext } { nested-horizontal }
5921 {
5922     The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ in ~ '#1' ~ 
5923 }
5924 \msg_new:nnn { enumext } { item-joined }
5925 {
5926     Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~\msg_line_context:.
5927 }
5928 \msg_new:nnn { enumext } { item-joined-columns }
5929 {
5930     Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.
5931 }

```

13.52 Finish package

Finish package implementation.

```
5932 \file_input_stop:
5933 </package>
```

14 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>	230
<code>\+</code>	222
<code>\-</code>	222
<code>\\</code>	238, 2921, 4342, 4345, 5550, 5559, 5564, 5584, 5586, 5593, 5595, 5608, 5613, 5618, 5633, 5672, 5674, 5676, 5681, 5682, 5687, 5688, 5706, 5723, 5740, 5745, 5754, 5763, 5769, 5775, 5784, 5789, 5798, 5812, 5822, 5832, 5842, 5852, 5862, 5868, 5874, 5883, 5888, 5893
A	
<code>above</code>	<u>1717</u>
<code>above*</code>	<u>1717</u>
<code>\addvspace</code>	1284, 1312, 1355, 1358, 1526, 1529, 1626, 1632, 1670, 1676, 1697, 1703, 3777, 3938, 3956, 4231, 4235, 4590, 4605, 4651, 4665
<code>after</code>	<u>1114</u>
<code>align</code>	667
<code>\Alph</code>	42, 46, 47
<code>\Alph</code>	609, 737, 781, 847, 5258
<code>\alph</code>	42, 46, 47
<code>\alph</code>	610, 735, 5250
<code>\anskey</code>	13, 80, 82, <u>2739</u>
<code>anskey*</code>	14, <u>2849</u>
<code>\anspic</code>	16, 110, 113, <u>4245</u>
<code>\anspic*</code>	74
<code>\arabic</code>	34, 42
<code>\arabic</code>	608, 734, 780, 5242, 5246, 5262
B	
<code>base-fix</code>	<u>976</u>
<code>\baselineskip</code>	55
<code>\baselineskip</code>	992, 999
<code>before</code>	<u>1114</u>
<code>before*</code>	<u>1114</u>
<code>below</code>	<u>1717</u>
<code>below*</code>	<u>1717</u>
bool commands:	
<code>\bool_gset_false:N</code>	351, 352, 353, 3025, 3027, 4607, 4611, 4667
<code>\bool_gset_true:N</code>	259, 269, 1217, 2210, 2216, 4576, 4608, 4640, 4668
<code>\bool_if:NTF</code>	402, 412, 429, 503, 510, 519, 526, 540, 553, 1739, 1753, 1766, 1777, 1788, 1799, 1810, 1821, 1870, 1887, 1892, 1900, 1927, 1965, 1970, 1977, 1981, 2003, 2008, 2016, 2023, 2054, 2062, 2155, 2371, 2381, 2460, 2484, 2491, 2515, 2613, 2635, 2675, 2689, 2693, 2743, 2762, 2786, 2840, 2851, 2940, 2977, 3041, 3076, 3091, 3166, 3266, 3300, 3336, 3352, 3373, 3509, 3530, 3618, 3628, 3661, 3666, 3732, 3758, 3808, 3866, 3921, 3946, 4164, 4229, 4247, 4266, 4313, 4340, 4569, 4585, 4591, 4634, 4648, 4652, 4741, 4751, 4839, 4845, 4852, 4868, 4961, 4971, 5088, 5093, 5122, 5138, 5169
<code>\bool_if:nTF</code>	1677, 1704, 3322, 3488, 3576, 4287, 5281, 5425
<code>\bool_if_p:N</code>	278, 293, 986, 987, 995, 996, 1649, 2034, 2035, 2043, 2044, 2168, 2194, 2207, 2208, 2213, 2214, 2548, 2558, 2570, 2585, 2586, 2620, 2661, 2662, 2963, 3153, 3154, 3183, 3184, 3222, 3223, 3242, 3243, 3522, 3523, 3524, 3705, 3707, 3718, 5114, 5115, 5116
<code>\bool_lazy_all:nTF</code>	276, 291, 984, 2166, 2192, 2546, 2555, 2568, 2583, 3220, 3240, 3520, 3703, 3716, 5112
<code>\bool_lazy_and:nnTF</code>	255, 265, 994, 1641, 1648, 2033, 2042, 2206, 2212, 2619, 2626, 2660, 2804, 2816, 2962, 2968, 3152
<code>\bool_lazy_or:nnTF</code>	2095, 2102, 3182, 5448
<code>\bool_new:N</code>	30, 31, 32, 33, 34, 35, 36, 37, 60, 69, 93, 98, 99, 104, 105, 108, 127, 137, 138, 145, 151, 152, 154, 158, 160, 161, 178, 190, 192
<code>\bool_not_p:n</code>	256, 266, 988, 1650, 2557, 2621, 2627, 2964, 2969, 3706, 3719
<code>\bool_set_eq:NN</code>	3275, 3469, 4792, 5036
<code>\bool_set_false:N</code>	409, 1006, 2140, 2141, 2173, 2178, 2182, 2186, 2199, 2904, 3497, 3680, 3825, 3874, 3961, 4103, 4161, 4303, 4710, 4736, 4789, 4977, 5032, 5033, 5294, 5295
<code>\bool_set_true:N</code>	283, 284, 298, 299, 394, 397, 660, 1021, 1723, 1728, 1990, 2112, 2113, 2403, 2411, 2905, 3269, 3271, 3303, 3305, 3465, 3476, 3490, 3641, 3679, 3712, 3725, 3798, 3871, 3898, 4100, 4289, 4290, 4558, 4623, 4709, 4796, 4803, 4804, 4848, 4975, 5040, 5047, 5048, 5049, 5289, 5290
box commands:	
<code>\box_dp:N</code>	1572, 1573, 1576, 1583, 1596, 1604, 1610, 1618, 4175, 4181, 4231, 4324
<code>\box_ht:N</code>	1355, 1358, 1369, 1370, 1381, 1383, 1398, 1401, 1409, 1410, 1421, 1423, 1438, 1441, 1448, 1449, 1460, 1462, 1477, 1480, 1526, 1529, 1537, 1538, 1546, 1547, 1559, 1561
<code>\box_ht_plus_dp:N</code>	4170, 4239, 4275
<code>\box_new:N</code>	66, 147, 148, 185, 191
<code>\box_use_drop:N</code>	4602, 4663, 4904, 5181
<code>\box_wd:N</code>	616
C	
<code>\c</code>	230, 231, 883, 885, 897, 899
<code>\catcode</code>	2921
<code>\cB</code>	231
<code>\cE</code>	231
<code>\centering</code>	1679, 1706, 4368, 4595, 4656
<code>check-ans</code>	<u>2132</u>
Document class:	
<code>article</code>	48
clist commands:	
<code>\clist_const:Nn</code>	197
<code>\clist_map_function:nN</code>	4351
<code>\clist_map_inline:Nn</code>	666, 931, 1113, 1128, 1209, 1733
<code>\clist_map_inline:nn</code>	45, 56, 74, 82, 95, 107, 140, 169, 196, 644, 697, 717, 1026, 1047, 1223, 1839, 2079, 2146, 2336, 2368, 2400, 2543, 3085, 3394, 3406, 3446, 3605, 3608, 3636, 3648, 3651, 3671, 5401
<code>\columnbreak</code>	81
<code>\columnbreak</code>	2623
<code>columns</code>	<u>1193</u>
<code>columns-sep</code>	<u>1193</u>
<code>\columnsep</code>	103
<code>\columnsep</code>	3753, 3919
<code>\columnseprule</code>	103
<code>\columnseprule</code>	3756, 3920

Commands provide by **enumext**:

\anskey 31, 70, 71, 76–80, 82, 83, 89, 102, 103, 122, 131, 132, 140
 \anspic* 31, 32, 74, 77, 90, 112, 113, 131, 132
 \anspic 31, 78, 110, 113, 140
 \foreachkeyans 136, 142
 \getkeyans 77, 131, 143
 \item* 31, 32, 74, 77, 78, 90, 93, 97, 123, 124, 129, 131, 132
 \item 93, 97, 117, 123, 125, 128
 \miniright 30, 53, 61, 62, 104, 105, 143
 \printkeyans* 132
 \printkeyans 31, 78, 132
 \setenumextmeta 135, 143
 \setenumext 31, 132, 134, 135, 138

Counters defined by **enumext**:

enumXiii 29, 41
 enumXii 29, 41
 enumXiv 29, 41
 enumXi 29, 41
 enumXviii 29, 41
 enumXvii 29, 41, 124
 enumXvi 29, 41
 enumXv 29, 41

cs commands:

\cs_generate_variant:Nn . 202, 203, 618, 634, 889, 905, 2452, 2457, 2533, 2857, 3595, 4353, 5460
 \cs_if_exist:NTF 588
 \cs_if_exist_p:N 3525, 5117
 \cs_if_free:NTF 2808, 2820
 \cs_new:Nn 216
 \cs_new:Npn . 234, 1840, 1849, 1857, 2415, 2424, 2432, 5309, 5318, 5327
 \cs_new_eq:NN . 378, 379, 384, 385, 414, 415, 418, 419
 \cs_new_protected:Nn . 226, 240, 248, 274, 307, 337, 343, 349, 355, 361, 369, 389, 437, 441, 459, 471, 489, 501, 517, 533, 546, 567, 757, 818, 869, 982, 1129, 1133, 1137, 1141, 1145, 1149, 1153, 1157, 1161, 1165, 1169, 1173, 1177, 1181, 1185, 1189, 1224, 1236, 1269, 1286, 1297, 1314, 1340, 1361, 1486, 1512, 1532, 1565, 1587, 1622, 1628, 1734, 1748, 1762, 1773, 1784, 1795, 1806, 1817, 1898, 2001, 2014, 2031, 2052, 2080, 2085, 2110, 2151, 2161, 2204, 2219, 2226, 2235, 2240, 2245, 2250, 2259, 2264, 2269, 2458, 2482, 2489, 2513, 2520, 2534, 2760, 2779, 2795, 2858, 2894, 2925, 2960, 3002, 3023, 3031, 3074, 3089, 3117, 3150, 3178, 3191, 3218, 3231, 3308, 3318, 3329, 3345, 3361, 3484, 3502, 3536, 3548, 3672, 3701, 3730, 3737, 3767, 3784, 3806, 3828, 3864, 3888, 3905, 3930, 3944, 3965, 4136, 4335, 4349, 4354, 4378, 4388, 4419, 4548, 4567, 4613, 4632, 4696, 4723, 4730, 4739, 4749, 4774, 4915, 4959, 4990, 4996, 5017, 5076, 5192
 \cs_new_protected:Npn 204, 208, 212, 422, 586, 603, 613, 619, 738, 782, 852, 876, 890, 1661, 1690, 1866, 1885, 1955, 1988, 2090, 2274, 2369, 2379, 2401, 2409, 2444, 2453, 2609, 2672, 2687, 2725, 2729, 2849, 2880, 2884, 2915, 3051, 3127, 3171, 3262, 3281, 3407, 3411, 3425, 3429, 3447, 3451, 3461, 3473, 3518, 3564, 3598, 3639, 3683, 3884, 4145, 4152, 4159, 4264, 4283, 4309, 4450, 4499, 4713, 4780, 4787, 4801, 4809, 4814, 4824, 4983, 5023, 5030, 5045, 5054, 5071, 5110, 5214, 5227, 5275, 5398, 5410, 5434, 5446, 5484, 5494, 5502, 5524
 \cs_new_protected_nopar:Nn . . . 4028, 4072, 4080, 4088, 4759, 4767, 4898, 5002, 5010, 5175
 \cs_new_protected_nopar:Npn . . 4020, 4036, 4830, 4876, 5128, 5149

\cs_set:Npn 2544, 2581, 5220
 \cs_set_eq:NN . . 3528, 4686, 4687, 4878, 4948, 4949, 5120, 5151
 \cs_set_protected:Nn 1052, 1068, 1081, 1093
 \cs_set_protected:Npn . 41, 50, 67, 75, 90, 96, 133, 165, 176, 635, 645, 667, 702, 718, 764, 906, 932, 1008, 1031, 1105, 1114, 1193, 1210, 1717, 1828, 2071, 2132, 2291, 2337, 2387, 2536, 3078, 3383, 3399, 3439, 3596, 3637
 \cs_to_str:N 605, 628
 \cs_undefine:N 2797, 2798, 2799, 2800

D

\d 222
 \DeclareDocumentEnvironment 571

dim commands:

\dim_abs:n 3569, 3574
 \dim_add:Nn 3216, 4179, 4413, 4444
 \dim_compare:nNnTF . . 1054, 1070, 1083, 1095, 1373, 1385, 1413, 1425, 1452, 1464, 1541, 1549, 1663, 1692, 3212, 3566, 3571, 3577, 3583, 3585, 3587, 3742, 3789, 3892, 3909, 4154, 4390, 4406, 4421, 4437, 4550, 4615, 5084
 \dim_compare:nTF 2645, 2990, 3831, 3968
 \dim_eval:n 992, 4237, 4320
 \dim_gset_eq:NN 4559, 4624
 \dim_gzero:N 3029, 4610, 4670
 \dim_new:N . 63, 70, 71, 72, 92, 129, 130, 142, 149, 150, 184, 186, 187, 193
 \dim_set:Nn . 616, 1022, 3199, 3203, 3208, 3214, 3298, 3569, 3574, 3576, 3579, 3580, 3584, 3586, 3589, 3590, 3592, 3745, 3792, 3830, 3894, 3911, 3967, 4168, 4273, 4356, 4392, 4399, 4423, 4430, 4485, 4534, 4552, 4617, 4826, 5086
 \dim_set_eq:NN 725, 771, 840, 844, 3293, 3607, 3650, 3753, 3919, 4492, 4495, 4496, 4541, 4544, 4545, 4819, 4890, 5163
 \dim_sub:Nn 3836, 3973, 4408, 4439
 \dim_use:N . 1055, 1063, 1664, 1674, 2523, 2526, 2531, 3313, 3315, 3368, 3743, 3747, 3748, 3750, 3790, 3795, 3796, 3802, 3833, 3838
 \dim_zero:N 3642, 3756, 3920, 4182
 \dim_zero_new:N 585
 \c_zero_dim 1057, 1071, 1084, 1096, 1664, 1692, 2647, 2992, 3199, 3212, 3566, 3571, 3577, 3584, 3743, 3790, 3833, 3892, 3909, 3970, 4154, 4390, 4406, 4421, 4437, 4550, 4615, 5084
 \dimeval 2298

E

\end . . . 2486, 2517, 3774, 3935, 4219, 4370, 5283, 5293, 5301
 end internal commands:

\end__enumext_mini_page . 1672, 1699, 3817, 3955, 4574, 4638, 4664

\endgroup 2921
 \endlist 379
 \endminipage 385
 enumext 5, 3842

enumext internal commands:

\l__enumext__ref_the_count_tl 44
 \l__enumext__resume_name_tl 66
 __enumext_add_meta_key:nnn . . . 136, 5412, 5428, 5429, 5431, 5434
 __enumext_add_pre_parsep: . 54, 1234, 1236, 1236
 __enumext_after_args_exec: 52, 1129, 1141, 3855
 __enumext_after_args_exec_v: 1145, 1157, 3988

```

\__enumext_after_args_exec_vii: .. 1161, 1185
\__enumext_after_args_exec_viii: ..... 1189
\__enumext_after_env:nn 86, 87, 89, 106, 119, 126,
    208, 208, 559, 563, 2935, 3860, 4583, 4646, 4931
\__enumext_after_hyperref: ... 37, 387, 387, 389
\l__enumext_after_list_args_v_tl ..... 1159
\l__enumext_after_list_args_vii_tl 1187, 4896
\l__enumext_after_list_args_viii_tl .. 1191,
    5173
\__enumext_after_list_vii: 119, 122, 4694, 4730,
    4730
\__enumext_after_list_viii: ... 127, 4957, 4996,
    4996
\__enumext_after_stop_list: 52, 105, 1129, 1137,
    3822
\__enumext_after_stop_list_v: 1145, 1153, 3962
\l__enumext_after_stop_list_v_tl ..... 1155
\__enumext_after_stop_list_vii: .. 122, 1161,
    1177, 4733
\l__enumext_after_stop_list_vii_tl ... 1179
\__enumext_after_stop_list_viii: . 1181, 4999
\l__enumext_after_stop_list_viii_tl ... 1183
\l__enumext_align_label_pos_v_str 3195, 3554
\l__enumext_align_label_pos_X_str ..... 75
\l__enumext_align_label_vii_str ..... 4865
\l__enumext_align_label_viii_str ..... 5142
\l__enumext_align_label_X_str ..... 176
\c__enumext_all_envs_clist . 197, 666, 931, 1113,
    1128, 1209, 1733
\c__enumext_all_families_seq .. 134, 5366, 5392
\l__enumext_anskey_env_bool 35, 85, 30, 284, 299,
    2851
\__enumext_anskey_env_clean_vars: . 88, 2956,
    2960, 3023
\__enumext_anskey_env_define_keys: 85, 2849,
    2858, 2929
\__enumext_anskey_env_exec: 87, 2854, 2925, 2925
\__enumext_anskey_env_make:n 70, 85, 2115, 2849,
    2849, 2857
\__enumext_anskey_env_reset_keys: 86, 87, 2849,
    2894, 2957
\__enumext_anskey_env_save_keys: .. 87, 2937,
    2960, 2960
\__enumext_anskey_env_store: .. 88, 2953, 2960,
    3002
\__enumext_anskey_env_unknown:n 86, 2849, 2877,
    2880
\__enumext_anskey_env_unknown:nn . 2849, 2882,
    2884
\l__enumext_anskey_level_int .. 24, 2781, 2782
\__enumext_anskey_safe_inner: . 84, 2754, 2760,
    2779
\__enumext_anskey_safe_inner:n ..... 83
\__enumext_anskey_safe_outer: . 83, 2741, 2760,
    2760
\__enumext_anskey_show_wrap_arg:n . 82, 2672,
    2672, 2691, 2706
\__enumext_anskey_show_wrap_left:n 82, 2617,
    2687, 2687
\__enumext_anskey_unknown:n 83, 2709, 2723, 2725
\__enumext_anskey_unknown:nn . 2709, 2727, 2729
\__enumext_anskey_wrapper:n ..... 2295, 2685
\l__enumext_anspic_above_int . 141, 4357, 4358,
    4360
\__enumext_anspic_args:nnn 113, 114, 4245, 4261,
    4335
\l__enumext_anspic_args_seq 113-115, 141, 4259,
    4369, 4382
\l__enumext_anspic_below_int . 141, 4357, 4358,
    4361
\l__enumext_anspic_body_box ... 141, 4272, 4275
\__enumext_anspic_body_dim:n .. 113, 4245, 4264,
    4312
\l__enumext_anspic_body_htdp_dim .. 113, 141,
    4273, 4323
__enumext_anspic_exec: ..... 4245
\__enumext_anspic_exec: ... 112, 115, 4214, 4378
\__enumext_anspic_label:nn 114, 4245, 4283, 4315,
    4330
\l__enumext_anspic_label_above_bool ... 141,
    4100, 4103, 4164, 4229, 4266, 4313, 4340
\l__enumext_anspic_label_box .. 141, 4167, 4170
\l__enumext_anspic_label_htdp_dim . 111, 141,
    4168, 4174, 4239, 4322
\__enumext_anspic_label_pos:nnn .. 114, 4245,
    4309, 4338
\l__enumext_anspic_label_sep_skip 4110, 4176,
    4240, 4325, 4342
\l__enumext_anspic_layout_style_tl 4112, 4380,
    4385
\l__enumext_anspic_mini_pos_str .. 141, 4101,
    4104, 4367
\l__enumext_anspic_mini_width_dim 141, 4285,
    4356, 4367
\__enumext_anspic_print:n 114, 115, 4245, 4349,
    4353, 4382, 4385
\__enumext_anspic_row:n .. 115, 4245, 4351, 4354
\__enumext_anspic_start_list_tag: 4044, 4072,
    4337
\__enumext_anspic_stop_list_tag: . 4044, 4088,
    4347
\__enumext_anspic_stop_start_list_tag: 4044,
    4080, 4339
\__enumext_at_begin_document:n .. 37, 204, 204,
    376, 382
\l__enumext_base_line_fix_bool 49, 133, 978, 987,
    1006, 5289, 5294
\__enumext_before_args_exec: 51, 104, 121, 1129,
    1129, 3787
\__enumext_before_args_exec_v: 1145, 1145, 3891
\__enumext_before_args_exec_vii: . 1161, 1161,
    4727
\__enumext_before_args_exec_viii: 1165, 4993
\__enumext_before_env:nn 85, 208, 212, 2802, 2814,
    2826, 2927
\__enumext_before_keys_exec: .. 51, 1129, 1133,
    3852
\__enumext_before_keys_exec_v: 1145, 1149, 3985
\__enumext_before_keys_exec_vii ..... 1161
\__enumext_before_keys_exec_vii: . 1169, 4681
\__enumext_before_keys_exec_viii: 1173, 4943
\__enumext_before_list: .. 104, 3784, 3784, 3846
\__enumext_before_list_v: ... 3888, 3888, 3980
\__enumext_before_list_vii: ... 121, 4676, 4723,
    4723
\__enumext_before_list_viii: .. 127, 4939, 4990,
    4990
\l__enumext_before_no_starred_key_v_tl 1151

```

`\l__enumext_before_no_starred_key_vii_-`
`tl 1171`
`\l__enumext_before_no_starred_key_viii_-`
`tl 1175`
`\l__enumext_before_starred_key_v_tl . . . 1147`
`\l__enumext_before_starred_key_vii_tl . 1163`
`\l__enumext_before_starred_key_viii_tl 1167`
`__enumext_calc_hspace:NNNNNN 99, 3564, 3564,`
`3595, 3600, 3643`
`__enumext_check_ans_active: 72, 104, 121, 2151,`
`2151, 3788, 4726`
`\g__enumext_check_ans_item_tl 91`
`\g__enumext_check_ans_key_bool . . 73, 151, 351,`
`2210, 2216, 3041`
`\l__enumext_check_ans_key_bool 73, 2136, 2141,`
`2207, 2213`
`__enumext_check_ans_key_hook: . . 73, 105, 122,`
`2204, 2204, 3823, 4734`
`__enumext_check_ans_level: 72, 2151, 2157, 2161`
`__enumext_check_ans_log: 73, 89, 2250, 2250, 3045`
`__enumext_check_ans_log_msg_greater: 2250,`
`2256, 2269`
`__enumext_check_ans_log_msg_less: 2250, 2254,`
`2259`
`__enumext_check_ans_log_msg_same_ok: 2250,`
`2255, 2264`
`__enumext_check_ans_msg_greater: 2226, 2232,`
`2245`
`__enumext_check_ans_msg_less: 2226, 2230, 2235`
`__enumext_check_ans_msg_same_ok: 2226, 2231,`
`2240`
`__enumext_check_ans_show: . . 73, 88, 2226, 2226,`
`3043`
`\l__enumext_check_answers_bool 70, 72, 83, 93, 94,`
`151, 2113, 2140, 2155, 2460, 2484, 2491, 2515, 2743,`
`2940, 3166, 3266, 3300, 4845`
`__enumext_check_starred_cmd:n 35, 74, 91, 126,`
`2274, 2274, 3991, 4227, 4956`
`\g__enumext_check_starred_cmd_int . . 98, 151,`
`2277, 2283, 2288, 3482, 4295, 5083`
`\l__enumext_check_start_line_env_tl . 35, 151,`
`314, 322, 330, 2280, 2286, 2289`
`\l__enumext_columns_sep_v_dim 3909, 3911, 3919`
`\l__enumext_columns_sep_vii_dim . . 4390, 4392,`
`4401, 4413, 4489, 4912`
`\l__enumext_columns_sep_viii_dim . 4421, 4423,`
`4432, 4444, 4538, 5189`
`\l__enumext_columns_v_int 1506, 1524, 1695, 3907,`
`3915, 3927, 3932`
`\l__enumext_columns_vii_int . . 4395, 4398, 4402,`
`4411, 4453, 4457, 4460, 4466, 4472, 4476, 4906, 4920`
`\l__enumext_columns_viii_int . 4426, 4429, 4433,`
`4442, 4502, 4506, 4509, 4515, 4521, 4525, 5183, 5198`
`\l__enumext_counter_i_tl 41, 595`
`\l__enumext_counter_ii_tl 41, 596`
`\l__enumext_counter_iii_tl 41, 597`
`\l__enumext_counter_iv_tl 41, 598`
`\c__enumext_counter_style_tl 34, 46, 228`
`\g__enumext_counter_styles_tl . 29, 42, 63, 606,`
`624`
`\l__enumext_counter_v_tl 41, 599, 860`
`\l__enumext_counter_vi_tl 41, 600`
`\l__enumext_counter_vii_tl 41, 601, 792`
`\l__enumext_counter_viii_tl 41, 602, 808`
`\l__enumext_current_widest_dim 29, 63, 630, 726,`
`772, 841, 845`
`__enumext_def_meta_key:nnn . . . 136, 5412, 5440,`
`5446, 5460`
`__enumext_default_item:n . . . 3262, 3262, 3326`
`__enumext_define_counters:Nn 29, 586, 586, 595,`
`596, 597, 598, 599, 600, 601, 602`
`__enumext_endminipage: . 37, 376, 385, 580, 4604,`
`4900, 5177`
`\g__enumext_envir_name_tl 35, 30, 285, 300, 359,`
`2083, 2088, 2098, 2238, 2243, 2248, 2262, 2267, 2272`
`\l__enumext_envir_name_tl . 34, 35, 30, 254, 264,`
`313, 321, 329, 5808, 5811, 5818, 5821, 5828, 5831,`
`5838, 5841, 5847, 5851, 5857, 5861, 5918, 5922`
`__enumext_execute_after_env: 36, 70, 73, 84, 88,`
`3031, 3031, 3862, 4933`
`__enumext_fake_item_indent: . 1052, 1052, 3627`
`\l__enumext_fake_item_indent_v_dim 1071, 1076`
`\l__enumext_fake_item_indent_v_tl 1073, 3466,`
`3470, 3477`
`__enumext_fake_item_indent_vii: . 1052, 1081,`
`3660`
`\l__enumext_fake_item_indent_vii_dim . 1084,`
`1088`
`\l__enumext_fake_item_indent_vii_tl . . 1086,`
`4895`
`__enumext_fake_item_indent_viii: 1052, 1093,`
`3665`
`\l__enumext_fake_item_indent_viii_dim 1096,`
`1100`
`\l__enumext_fake_item_indent_viii_tl . 1098,`
`5168`
`\l__enumext_fake_item_indent_X_tl 96`
`__enumext_fake_make_label_vii:n . 124, 4830,`
`4830, 4887`
`__enumext_fake_make_label_viii:n 5110, 5128,`
`5160`
`__enumext_filter_first_level:n . . 133, 5309,`
`5309, 5343, 5354`
`__enumext_filter_first_level_key:n 134, 5309,`
`5314, 5318`
`__enumext_filter_first_level_pair:nn . 134,`
`5309, 5315, 5327`
`__enumext_filter_save_key:n . . 77, 2376, 2384,`
`2407, 2413, 2415, 5240, 5244, 5248, 5252, 5256,`
`5260`
`__enumext_filter_save_key_key:n . . 77, 2415,`
`2420, 2424`
`__enumext_filter_save_key_pair:nn 77, 2415,`
`2421, 2432`
`__enumext_filter_series:n 65, 1840, 1840, 1878,`
`1890, 1895`
`__enumext_filter_series_key:n 65, 1840, 1845,`
`1849`
`__enumext_filter_series_pair:nn . . 65, 1840,`
`1846, 1857`
`__enumext_first_item_tmp_vii: 120, 122, 4686,`
`4759, 4759`
`__enumext_first_item_tmp_viii: . . 128, 4948,`
`5002, 5002`
`\g__enumext_footnote_standar_arg_seq . . 170,`
`454, 465, 468`
`\g__enumext_footnote_standar_int 170, 448, 451,`
`453, 456`
`\g__enumext_footnote_standar_int_seq . . 170,`
`456, 461, 464, 469`

\g__enumext_footnote_starred_arg_seq .. 170, 484, 495, 498
 \g__enumext_footnote_starred_int 170, 478, 481, 483, 486
 \g__enumext_footnote_starred_int_seq .. 170, 486, 491, 494, 499
 __enumext_footnotes_key_bool 37
 \l__enumext_footnotes_key_bool 32, 37, 160, 397, 402, 409, 510, 526, 540, 553
 __enumext_footnotetext:nn .. 437, 437, 466, 496
 __enumext_foreach_add_body:n . 137, 5461, 5521, 5524
 \l__enumext_foreach_after_tl 5465, 5533
 \l__enumext_foreach_before_tl 5463, 5528
 \g__enumext_foreach_default_keys_tl 136, 122, 5483, 5504
 __enumext_foreach_keyans:nn .. 137, 5461, 5500, 5502
 \l__enumext_foreach_name_prop_tl . 122, 5506, 5531
 \l__enumext_foreach_print_seq 122, 5516, 5522, 5526
 \l__enumext_foreach_sep_tl 5475, 5522
 \l__enumext_foreach_start_int 5467, 5518
 \l__enumext_foreach_step_int 5471, 5519
 \l__enumext_foreach_stop_int . 5469, 5511, 5513, 5520
 __enumext_foreach_wrapper:n 5473, 5529
 __enumext_getkeyans:nn .. 132, 5209, 5223, 5227
 __enumext_getkeyans_aux:n 131, 5209, 5211, 5214
 \l__enumext_hyperref_bool 32, 37, 160, 394, 412, 429, 2662, 3154, 4839
 __enumext_hypertarget:nn 37, 387, 414, 418, 434
 __enumext_if_is_int:n 220
 __enumext_if_is_int:nTF 220, 878, 892
 __enumext_internal_mini_page: 40, 102, 121, 567, 567, 3675, 4699
 __enumext_is_not_nested: . 29, 34, 102, 121, 248, 248, 3674, 4698
 __enumext_is_on_first_level: . 29, 35, 102, 121, 248, 274, 3681, 4711
 \g__enumext_item_anskey_int 83, 91, 151, 346, 373, 374, 2223, 2611, 3168
 __enumext_item_answer_diff: 73, 88, 2219, 2219, 3038
 \g__enumext_item_answer_diff_int 73, 151, 347, 2221, 2228, 2252
 \l__enumext_item_column_pos_vii_int 122, 4460, 4466, 4472, 4476, 4483, 4770, 4906, 4909
 \l__enumext_item_column_pos_viii_int .. 128, 4509, 4515, 4521, 4525, 4532, 5013, 5183, 5186
 \l__enumext_item_column_pos_X_int 176
 \g__enumext_item_count_all_vii_int 122, 4484, 4771, 4920, 4928
 \g__enumext_item_count_all_viii_int 128, 4533, 5014, 5197, 5206
 \g__enumext_item_count_all_X_int 176
 \g__enumext_item_number_bool 151
 \l__enumext_item_number_bool 72, 158, 2173, 2178, 2182, 2186, 2199, 2786, 2840, 3269, 3303, 4848
 \g__enumext_item_number_int .. 72, 151, 345, 372, 374, 2172, 2177, 2181, 2185, 2198, 2223, 3268, 3302, 4847
 __enumext_item_peek_args_vii: 123, 4767, 4772, 4774
 __enumext_item_peek_args_viii: .. 128, 5010, 5015, 5017
 \l__enumext_item_starred_vii_bool 4789, 4803, 4852
 \l__enumext_item_starred_viii_bool 5032, 5047, 5138, 5169
 \l__enumext_item_starred_X_bool 176
 __enumext_item_std:w 37, 93, 94, 97, 376, 380, 3272, 3278, 3306, 3466, 3470, 3477
 \g__enumext_item_symbol_aux_tl . 94, 126, 3286, 3289, 3314, 3358, 3378
 \g__enumext_item_symbol_aux_vii_tl 4811, 4854, 4857, 4861, 4863
 \g__enumext_item_symbol_aux_X_tl 176
 \l__enumext_item_symbol_sep_vii_dim .. 4819, 4826, 4860, 4862
 \l__enumext_item_symbol_vii_tl 4857
 \l__enumext_item_text_vii_box 4879, 4904
 \l__enumext_item_text_viii_box ... 5152, 5181
 \l__enumext_item_text_X_box 176
 \l__enumext_item_width_vii_dim ... 4399, 4408, 4487, 4495, 4496
 \l__enumext_item_width_viii_dim .. 4430, 4439, 4536, 4544, 4545
 \l__enumext_item_width_X_dim 176
 \l__enumext_item_wrap_key_bool . 98, 151, 3223, 3243, 3490, 3497, 3524, 4289, 4303, 5033, 5048, 5116
 \l__enumext_itemindent_X_dim 67
 \l__enumext_itemsep_i_skip ... 1367, 1374, 1377, 1379, 1386, 1390, 1393, 1395, 1535, 1542, 1544, 1545, 1550, 1554, 1556, 1557
 \l__enumext_itemsep_ii_skip .. 1407, 1414, 1417, 1419, 1426, 1430, 1433, 1435
 \l__enumext_itemsep_iii_skip . 1446, 1453, 1456, 1458, 1465, 1469, 1472, 1474
 \l__enumext_itemsep_vii_skip 4926
 \l__enumext_itemsep_viii_skip 5204
 \l__enumext_joined_item_aux_vii_int .. 4481, 4482, 4483, 4484, 4490
 \l__enumext_joined_item_aux_viii_int . 4530, 4531, 4532, 4533, 4539
 \l__enumext_joined_item_aux_X_int 176
 __enumext_joined_item_vii:w .. 123, 4767, 4777, 4778, 4780
 \l__enumext_joined_item_vii_int .. 4452, 4453, 4456, 4458, 4464, 4469, 4474, 4479, 4481, 4487
 __enumext_joined_item_viii:w . 128, 5010, 5020, 5021, 5023
 \l__enumext_joined_item_viii_int . 4501, 4502, 4505, 4507, 4513, 4518, 4523, 4528, 4530, 4536
 \l__enumext_joined_item_X_int 176
 \l__enumext_joined_width_vii_dim . 4485, 4492, 4495, 4881, 4889
 \l__enumext_joined_width_viii_dim 4534, 4541, 4544, 5154, 5162
 \l__enumext_joined_width_X_dim 176
 __enumext_keyans_addto_prop:n 89, 3051, 3051, 3479, 4292
 __enumext_keyans_addto_seq:n . 90, 3127, 3127, 3481, 4294
 __enumext_keyans_addto_seq_link: 3127, 3148, 3150, 5082
 __enumext_keyans_default_item:n .. 97, 3461, 3461, 3498

`\l__enumext_keyans_env_bool` [30](#), [3706](#), [3719](#), [3871](#), [3961](#)
`__enumext_keyans_fake_item_indent` : [1052](#), [1068](#), [3617](#)
`\l__enumext_keyans_level_h_int` : [126](#), [24](#), [801](#), [827](#), [2770](#), [2832](#), [3105](#), [4705](#), [4965](#), [4966](#)
`\l__enumext_keyans_level_int` : [24](#), [1655](#), [2766](#), [2828](#), [3100](#), [3233](#), [3870](#), [3875](#), [4255](#)
`__enumext_keyans_make_label` : [98](#), [3502](#), [3502](#), [3615](#)
`__enumext_keyans_make_label_box` : [3502](#), [3506](#), [3511](#), [3548](#)
`__enumext_keyans_make_label_std` : [3502](#), [3514](#), [3536](#)
`__enumext_keyans_mini_right_cmd:n` [62](#), [1657](#), [1690](#), [1690](#)
`__enumext_keyans_mini_set_vskip` : [58](#)
`__enumext_keyans_minipage_add_space` : [1486](#), [1512](#), [3900](#)
`__enumext_keyans_minipage_set_skip` : [1486](#), [1486](#), [1514](#)
`__enumext_keyans_multi_addvspace` : [1286](#), [1297](#), [3924](#)
`__enumext_keyans_multi_set_vskip` : [55](#), [1286](#), [1286](#), [1299](#)
`__enumext_keyans_multicols_start` : [3888](#), [3903](#), [3905](#)
`__enumext_keyans_multicols_stop` : [1694](#), [3888](#), [3930](#), [3959](#)
`__enumext_keyans_name_and_start` : [29](#), [35](#), [126](#), [307](#), [307](#), [3872](#), [4143](#), [4970](#)
`__enumext_keyans_parse_keys:n` [3884](#), [3884](#), [3979](#)
`__enumext_keyans_pic_arg_two` : [111](#), [4136](#), [4159](#), [4190](#)
`\l__enumext_keyans_pic_level_int` : [24](#), [1636](#), [2774](#), [2836](#), [3054](#), [3095](#), [3130](#), [4138](#), [4139](#)
`__enumext_keyans_pic_parse_keys:n` [4136](#), [4145](#), [4189](#)
`__enumext_keyans_pic_safe_exec` : [111](#), [4136](#), [4136](#), [4188](#)
`__enumext_keyans_pic_skip_abs:N` : [111](#), [4136](#), [4152](#), [4163](#)
`__enumext_keyans_pos_mark_set` : [92](#), [3191](#), [3191](#), [3226](#), [3257](#)
`__enumext_keyans_pre_itemsep_skip` : [1486](#), [1505](#), [1532](#)
`__enumext_keyans_redefine_item` : [98](#), [3484](#), [3484](#), [3614](#)
`__enumext_keyans_ref` : [46](#), [852](#), [869](#), [3616](#)
`__enumext_keyans_ref:n` : [46](#), [849](#), [852](#), [852](#)
`__enumext_keyans_safe_exec` : [3864](#), [3864](#), [3978](#)
`__enumext_keyans_save_item_opt:n` [91](#), [97](#), [3171](#), [3171](#), [3475](#), [4291](#)
`__enumext_keyans_set_item_width` : [107](#), [3965](#), [3965](#), [3987](#)
`__enumext_keyans_show_ans` : [92](#), [3191](#), [3218](#), [3541](#), [3556](#), [4296](#)
`__enumext_keyans_show_item_opt` : [91](#), [97](#), [3171](#), [3178](#), [3478](#), [4300](#), [5171](#)
`__enumext_keyans_show_pos` : [92](#), [3191](#), [3231](#), [3542](#), [3557](#), [4297](#)
`__enumext_keyans_starred_item:n` : [97](#), [3473](#), [3473](#), [3493](#)
`__enumext_keyans_starred_item_star` : [129](#), [5045](#), [5076](#), [5140](#)
`__enumext_keyans_store_ref` : [90](#), [3074](#), [3074](#), [3480](#), [4293](#), [5080](#)
`__enumext_keyans_store_ref_aux_i` : [90](#), [3074](#), [3086](#), [3089](#)
`__enumext_keyans_store_ref_aux_ii` : [90](#), [3074](#), [3115](#), [3117](#)
`__enumext_keyans_unknown_keys:n` : [3399](#), [3403](#), [3407](#), [4134](#)
`__enumext_keyans_unknown_keys:nn` [3399](#), [3409](#), [3411](#)
`__enumext_keyans_wrapper_label:n` : [98](#)
`__enumext_keyans_wrapper_label_viii:n` [5110](#), [5110](#), [5145](#)
`__enumext_keyans_wrapper_item:n` : [2304](#), [2358](#), [3525](#), [3528](#), [5117](#), [5120](#)
`__enumext_keyans_wrapper_label:n` [3502](#), [3518](#), [3544](#), [3559](#), [4299](#)
`__enumext_keyans_wrapper_opt:n` : [2301](#), [2356](#), [3186](#)
`\l__enumext_label_copy_i_tl` : [2577](#), [3093](#), [3098](#), [3103](#), [3108](#)
`\l__enumext_label_copy_v_tl` : [3103](#)
`\l__enumext_label_copy_vi_tl` : [3098](#)
`\l__enumext_label_copy_vii_tl` [2553](#), [2564](#), [2593](#), [3093](#)
`\l__enumext_label_copy_viii_tl` : [3108](#)
`\l__enumext_label_copy_X_tl` : [162](#)
`\l__enumext_label_fill_left_v_tl` : [3540](#)
`\l__enumext_label_fill_left_X_tl` : [96](#)
`\l__enumext_label_fill_right_v_tl` : [3545](#)
`\l__enumext_label_fill_right_X_tl` : [96](#)
`\l__enumext_label_font_style_v_tl` [3543](#), [3558](#), [4298](#), [4304](#)
`\l__enumext_label_font_style_vii_tl` : [4867](#)
`\l__enumext_label_font_style_viii_tl` : [5144](#)
`\l__enumext_label_i_tl` : [718](#)
`\l__enumext_label_ii_tl` : [718](#)
`\l__enumext_label_iii_tl` : [718](#)
`\l__enumext_label_iv_tl` : [718](#)
`__enumext_label_style:Nnn` [29](#), [42](#), [619](#), [619](#), [634](#), [723](#), [769](#), [838](#), [842](#)
`\l__enumext_label_v_tl` [90](#), [835](#), [3059](#), [3135](#), [3194](#), [3982](#), [4167](#)
`\l__enumext_label_vi_tl` [90](#), [835](#), [3056](#), [3132](#), [4299](#), [4305](#)
`\l__enumext_label_vii_tl` : [764](#), [4798](#), [4821](#), [4828](#)
`\l__enumext_label_viii_tl` [764](#), [5042](#), [5074](#), [5078](#)
`\l__enumext_label_width_by_box` : [63](#), [615](#), [616](#)
`__enumext_label_width_by_box:Nn` [42](#), [613](#), [613](#), [618](#), [630](#), [902](#), [3193](#)
`\l__enumext_labelsep_v_dim` : [3214](#), [3914](#), [4179](#)
`\l__enumext_labelsep_vii_dim` : [2678](#), [4394](#), [4404](#), [4488](#), [4763](#), [4819](#), [4874](#), [4883](#)
`\l__enumext_labelsep_viii_dim` [4425](#), [4435](#), [4537](#), [5006](#), [5086](#), [5147](#), [5156](#)
`\l__enumext_labelwidth_v_dim` : [3204](#), [3209](#), [3228](#), [3259](#), [3554](#), [3914](#), [4179](#)
`\l__enumext_labelwidth_vii_dim` : [2678](#), [4394](#), [4403](#), [4488](#), [4763](#), [4865](#), [4882](#)
`\l__enumext_labelwidth_viii_dim` : [4425](#), [4434](#), [4537](#), [5006](#), [5091](#), [5107](#), [5142](#), [5155](#)
`\l__enumext_leftmargin_tmp_v_bool` : [111](#), [4161](#)
`\l__enumext_leftmargin_tmp_X_bool` : [67](#)
`\l__enumext_leftmargin_tmp_X_dim` : [67](#)

```

\l__enumext_leftmargin_X_dim . . . . . 67
\__enumext_level: 216, 216, 747, 750, 751, 759, 761,
1055, 1059, 1063, 1131, 1135, 1139, 1143, 1226, 1228,
1230, 1232, 1274, 1276, 1278, 1280, 1284, 1318, 1324,
1329, 1331, 1334, 1337, 1350, 1353, 1664, 1668, 1674,
1737, 1739, 1741, 1744, 1751, 1753, 1755, 1758, 2371,
2373, 2375, 2403, 2404, 2406, 2462, 2470, 2474, 2478,
2682, 2683, 3271, 3272, 3276, 3277, 3278, 3286, 3294,
3295, 3298, 3305, 3306, 3310, 3313, 3315, 3349, 3351,
3352, 3354, 3357, 3368, 3369, 3372, 3373, 3375, 3712,
3725, 3732, 3740, 3743, 3745, 3747, 3748, 3749, 3750,
3753, 3758, 3764, 3770, 3777, 3790, 3792, 3795, 3796,
3798, 3802, 3808, 3833, 3838, 3849, 3851
\l__enumext_level_h_int 121, 24, 257, 280, 294, 785,
820, 1643, 2169, 2189, 2572, 2806, 2818, 3720, 4700,
4701
\l__enumext_level_int . 102, 24, 218, 267, 279, 295,
569, 1238, 1363, 1642, 2163, 2195, 2549, 2559, 2565,
2571, 2578, 2587, 2592, 2805, 2817, 3033, 3631, 3676,
3677, 3688, 3696, 3710, 3723, 3754, 3879, 4251, 4743,
4753, 4978, 5848, 5852, 5858, 5862
\__enumext_list_arg_two_i: . . . . . 3596
\__enumext_list_arg_two_ii: . . . . . 3596
\__enumext_list_arg_two_iii: . . . . . 3596
\__enumext_list_arg_two_iv: . . . . . 3596
\__enumext_list_arg_two_v: . 98, 3596, 3984, 4162
\__enumext_list_arg_two_vii: . . . . 3637, 4680
\__enumext_list_arg_two_viii: . . . . 3637, 4942
\l__enumext_listoffset_v_dim . 3916, 3970, 3973
\l__enumext_listparindent_vii_dim 4890, 4894
\l__enumext_listparindent_viii_dim 5163, 5167
\__enumext_log_answer_vars: . 36, 361, 369, 3040
\__enumext_log_global_vars: . 36, 361, 361, 3039
\__enumext_make_label: . . . . 95, 3329, 3329, 3625
\__enumext_make_label_box: . . . 3329, 3333, 3338,
3361
\__enumext_make_label_std: . . . 3329, 3341, 3345
\l__enumext_mark_answer_sym_tl 79, 2309, 2341,
2528, 2695, 3246, 5095
\l__enumext_mark_position_str 126, 2313, 2314,
2344, 2345, 2526
\l__enumext_mark_ref_sym_tl . 2328, 2667, 3162
\l__enumext_mark_sep_tmpa_dim 126, 3194, 3204,
3209
\l__enumext_mark_sep_tmpb_dim 126, 3199, 3203,
3208, 3216
\l__enumext_mark_sym_sep_dim . 2320, 2352, 3212,
3214, 3216, 3228, 3259, 5084, 5086, 5091, 5107
\l__enumext_meta_path_tl . 122, 5436, 5437, 5439,
5440
\c__enumext_meta_paths_prop . . . . . 135, 5412
\__enumext_mini_addvspace_vii: 61, 1622, 1622,
4562
\__enumext_mini_addvspace_viii: 61, 1622, 1628,
4627
__enumext_mini_env* . . . . . 567
\__enumext_mini_page 1674, 1701, 3802, 3901, 4564,
4629, 4650
\__enumext_mini_right_cmd:n . 61, 62, 1659, 1661,
1661
\__enumext_mini_set_vskip_vii: 60, 1565, 1565,
1624
\__enumext_mini_set_vskip_viii: 60, 1565, 1587,
1630
\__enumext_minipage:w 37, 376, 384, 574, 4587, 4889,
5162
\l__enumext_minipage_active_v_bool 3898, 3921,
3946
\g__enumext_minipage_active_vii_bool . . 119,
4576, 4585, 4607
\l__enumext_minipage_active_vii_bool . 4558,
4569
\g__enumext_minipage_active_viii_bool 4640,
4648, 4667
\l__enumext_minipage_active_viii_bool 4623,
4634
\g__enumext_minipage_active_X_bool . . . 176
\l__enumext_minipage_active_X_bool . . . . 83
\__enumext_minipage_add_space: . 56, 104, 1314,
1340, 3800
\g__enumext_minipage_after_skip 83, 1569, 1581,
4605, 4665
\l__enumext_minipage_after_skip . . 56, 105, 83,
1327, 1367, 1369, 1374, 1377, 1381, 1386, 1390, 1393,
1397, 1409, 1414, 1417, 1421, 1426, 1430, 1433, 1437,
1448, 1453, 1456, 1460, 1465, 1469, 1472, 1476, 1488,
1502, 1535, 1537, 1542, 1544, 1546, 1550, 1554, 1556,
1558, 1589, 1602, 1616, 1670, 1697, 3956
\g__enumext_minipage_center_vii_bool . 4591,
4608
\g__enumext_minipage_center_viii_bool 4652,
4668
\g__enumext_minipage_center_X_bool . . . 176
\l__enumext_minipage_hsep_v_dim . . . . . 3896
\l__enumext_minipage_hsep_vii_dim . . . . 4556
\l__enumext_minipage_hsep_viii_dim . . . 4621
\l__enumext_minipage_left_skip 83, 1489, 1567,
1572, 1576, 1590, 1594, 1608, 1626, 1632
\l__enumext_minipage_left_v_dim . . 3894, 3901
\l__enumext_minipage_left_vii_dim 4552, 4564
\l__enumext_minipage_left_viii_dim 4617, 4629
\l__enumext_minipage_left_X_dim . . . . . 83
\g__enumext_minipage_right_skip 83, 1568, 1573,
1577, 4590, 4651
\l__enumext_minipage_right_skip . 56, 83, 1316,
1322, 1327, 1329, 1331, 1490, 1491, 1497, 1502, 1503,
1504, 1509, 1591, 1598, 1612, 1676, 1703
\l__enumext_minipage_right_v_dim . 1692, 1701,
3892, 3896
\g__enumext_minipage_right_vii_dim 118, 4560,
4587, 4610
\l__enumext_minipage_right_vii_dim 118, 4550,
4555, 4561
\g__enumext_minipage_right_viii_dim . . 4625,
4650, 4670
\l__enumext_minipage_right_viii_dim . . 4615,
4620, 4626
\g__enumext_minipage_right_X_dim . . . . . 176
\g__enumext_minipage_right_X_skip . . . . 176
\__enumext_minipage_set_skip: . 56, 1314, 1314,
1342
\g__enumext_minipage_stat_int . . 104, 83, 1681,
1708, 3799, 3810, 3815, 3899, 3948, 3953
\l__enumext_minipage_temp_skip 83, 1388, 1398,
1401, 1428, 1438, 1441, 1467, 1477, 1480, 1552, 1559,
1561
\l__enumext_miniright_code_vii_box 4598, 4602
\g__enumext_miniright_code_vii_tl 119, 4593,
4600, 4609

```

`\l__enumext_miniright_code_viii_box` .. 4659, 4663
`\g__enumext_miniright_code_viii_tl` 4654, 4661, 4669
`\l__enumext_miniright_code_X_box` 176
`\l__enumext_mode_box_bool` 639, 3336, 3509
`__enumext_multi_addvspace`: 54, 103, 1269, 1269, 3761
`__enumext_multi_set_vskip`: 54, 1224, 1224, 1271
`\l__enumext_multicols_above_ii_skip` ... 1243
`\l__enumext_multicols_above_iii_skip` .. 1252
`\l__enumext_multicols_above_iv_skip` ... 1261
`\l__enumext_multicols_above_v_skip` 1288, 1302, 1312, 1503
`\l__enumext_multicols_above_X_skip` 75
`\l__enumext_multicols_below_ii_skip` .. 1370, 1379, 1383, 1395, 1400
`\l__enumext_multicols_below_iii_skip` . 1410, 1419, 1423, 1435, 1440
`\l__enumext_multicols_below_iv_skip` .. 1449, 1458, 1462, 1474, 1479
`\l__enumext_multicols_below_v_skip` 1292, 1306, 1504, 1538, 1545, 1547, 1557, 1560, 3938
`\l__enumext_multicols_below_X_skip` 75
`\g__enumext_multicols_right_X_skip` 75
`__enumext_multicols_start`: 103, 104, 3737, 3737, 3804
`__enumext_multicols_stop`: 104, 1666, 3767, 3767, 3820
`__enumext_nested_base_line_fix`: 49, 102, 976, 982, 3692
`__enumext_newlabel:nn` 32, 38, 80, 422, 422, 2603, 3121
`\l__enumext_newlabel_arg_one_tl` 32, 38, 80, 90, 162, 2596, 2604, 2666, 3110, 3122, 3160
`\l__enumext_newlabel_arg_two_tl` 32, 38, 79, 162, 2552, 2562, 2575, 2590, 2605, 3097, 3102, 3107, 3123
`__enumext_parse_foreach_keys:n` .. 5461, 5477, 5494
`__enumext_parse_foreach_keys:nn` . 5461, 5484, 5496
`__enumext_parse_keys:n` 49, 65, 3683, 3683, 3845
`__enumext_parse_keys_vii:n` 65, 4675, 4713, 4713
`__enumext_parse_keys_viii:n` . 4938, 4983, 4983
`__enumext_parse_save_key:n` 76, 2396, 2401, 2401
`__enumext_parse_save_key_vii:n` 76, 2391, 2401, 2409
`__enumext_parse_series:n` .. 65, 102, 121, 1866, 1866, 3691, 4719
`__enumext_parse_store_keys:n` 102
`\l__enumext_parsep_i_skip` 1241, 1245
`\l__enumext_parsep_ii_skip` 1250, 1254
`\l__enumext_parsep_iii_skip` 1259, 1263
`\l__enumext_parsep_vii_skip` 4891
`\l__enumext_parsep_viii_skip` 5164
`\l__enumext_partopsep_v_skip` . 1304, 1308, 1499, 1522
`\l__enumext_partopsep_viii_skip` 1600
`__enumext_phantomsection`: 37, 387, 415, 419, 435
`__enumext_pre_itemsep_skip`: 56, 57, 1332, 1361, 1361
`__enumext_print_footnote`: .. 437, 459, 523, 528
`__enumext_print_footnote_mini`: 437, 489, 550, 555
`__enumext_print_footnote_standar`: 501, 517, 581
`__enumext_print_footnote_starred`: 501, 546, 561, 565
`__enumext_print_keyans_box:NN` 79, 2520, 2520, 2533, 2677, 2681, 3227, 3258, 5090, 5106
`\l__enumext_print_keyans_i_tl` 5245, 5267
`\l__enumext_print_keyans_ii_tl` ... 5249, 5268
`\l__enumext_print_keyans_iii_tl` .. 5253, 5269
`\l__enumext_print_keyans_iv_tl` ... 5257, 5270
`\l__enumext_print_keyans_star_bool` . 49, 133, 126, 988, 996, 5290, 5295
`\l__enumext_print_keyans_starred_tl` 132, 133, 126, 5241, 5288
`\l__enumext_print_keyans_vii_tl` 132, 5261, 5271
`\l__enumext_print_keyans_X_tl` 126
`__enumext_printkeyans:nnn` 132, 133, 5264, 5272, 5275
`__enumext_redefine_item`: . 94, 3318, 3318, 3624
`\l__enumext_ref_key_arg_tl` 44, 46, 231, 740, 741, 753, 784, 787, 797, 803, 813, 854, 855, 865
`\l__enumext_ref_the_count_tl` . 44, 46, 747, 750, 753, 792, 794, 797, 808, 810, 813, 860, 862, 865
`__enumext_regex_counter_style`: .. 34, 44, 226, 226, 748, 793, 809, 861
`__enumext_register_counter_style:Nn` .. 603, 603, 608, 609, 610, 611, 612
`__enumext_remove_extra_parsep_vii`: .. 4693, 4915, 4915
`__enumext_remove_extra_parsep_viii`: . 4955, 5192, 5192
`__enumext_renew_footnote`: .. 437, 441, 507, 512
`__enumext_renew_footnote_mini`: 437, 471, 537, 542
`__enumext_renew_footnote_standar`: 501, 501, 573
`__enumext_renew_footnote_starred`: 501, 533, 4885, 5158
`\l__enumext_renew_the_count_v_tl` 863, 871, 873
`\l__enumext_renew_the_count_vii_tl` 795, 822, 824
`\l__enumext_renew_the_count_viii_tl` 811, 829, 831
`\l__enumext_renew_the_count_X_tl` 46
`__enumext_rescan_anskey_env:n` .. 86, 88, 2849, 2915, 3010, 3018
`__enumext_reset_global_bool`: .. 337, 340, 349
`__enumext_reset_global_int`: ... 337, 339, 343
`__enumext_reset_global_tl`: 337, 341, 355
`__enumext_reset_global_vars`: . 36, 89, 337, 337, 3048
`\l__enumext_resume_active_bool` 65, 68, 57, 1870, 1990
`__enumext_resume_counter`: .. 67, 68, 1988, 1994, 2001
`__enumext_resume_counter:n` . 65, 68, 1959, 1964, 1988, 1988, 2058, 2066
`__enumext_resume_counter_save_ans`: .. 68, 69, 1988, 1999, 2031
`__enumext_resume_counter_series`: . 68, 1988, 1997, 2014
`\g__enumext_resume_int` ... 57, 1911, 2005, 2006
`__enumext_resume_last:n` 65, 66, 1866, 1872, 1885
`\l__enumext_resume_name_tl` 57, 1907, 1915, 1918,

1934, 1942, 1945, 1991, 1992, 2020, 2027
 __enumext_resume_save_counter: . 66, 105, 122, 1898, 1898, 3826, 4737
 __enumext_resume_series:n . 67, 1834, 1955, 1955
 __enumext_resume_starred: . 69, 1835, 2052, 2052
 \g__enumext_resume_vii_int 57, 1938, 2010, 2011
 \l__enumext_rightmargin_vii_dim . 4406, 4410, 4415
 \l__enumext_rightmargin_viii_dim . 4437, 4441, 4446
 __enumext_safe_exec: . . 40, 102, 3672, 3672, 3844
 __enumext_safe_exec_vii: . 40, 4674, 4696, 4696
 __enumext_safe_exec_viii: 126, 4937, 4959, 4959
 __enumext_second_part: . . 105, 3806, 3806, 3858
 __enumext_second_part_v: . . . 3888, 3944, 3992
 \l__enumext_series_name_tl 68
 \l__enumext_series_str . 66, 102, 121, 1832, 1868, 1876, 1877, 1879, 1881, 1902, 1905, 1909, 1929, 1932, 1936, 3687, 4717
 __enumext_set_error:nn 5371, 5408, 5410
 __enumext_set_item_width: 105, 3828, 3828, 3854
 __enumext_set_parse:n 5371, 5382, 5398
 \l__enumext_setkey_tmpa_int . . . 117, 5375, 5379
 \l__enumext_setkey_tmpa_seq . . 117, 5373, 5383, 5389, 5391, 5393, 5405
 \l__enumext_setkey_tmpa_tl . . . 117, 5381, 5385
 \l__enumext_setkey_tmpb_seq . . 117, 5374, 5377, 5381, 5382
 \l__enumext_setkey_tmpb_tl 117, 5400, 5402, 5403
 \l__enumext_show_answer_bool . 2322, 2360, 2689, 3183, 3222, 3523, 5088, 5115
 __enumext_show_length:nnn . . 51, 234, 234, 5619, 5620, 5621, 5622, 5623, 5624, 5625, 5626, 5627, 5628, 5634, 5635, 5636, 5637, 5638, 5639, 5640, 5641, 5642, 5643
 \l__enumext_show_pos_tmp_int . 126, 3235, 3238, 3252
 \l__enumext_show_position_bool . . . 2325, 2363, 2693, 3184, 3242, 5093
 \g__enumext_standar_bool 34, 102, 30, 256, 259, 278, 352, 503, 519, 1900, 1965, 1977, 2003, 2016, 2054, 2194, 2208, 2557, 2570, 2585, 3707
 \l__enumext_standar_bool 102, 105, 30, 1650, 2558, 3679, 3825, 4710
 \l__enumext_standar_first_bool 35, 102, 30, 283, 1887, 2034, 2096, 2103
 __enumext_standar_item_vii:w . 123, 4767, 4785, 4787
 __enumext_standar_item_viii:w 128, 5010, 5028, 5030
 __enumext_standar_ref: 44, 738, 757, 3626
 __enumext_standar_ref:n 44, 730, 738, 738
 \g__enumext_standar_series_tl . 57, 1889, 1890, 2056, 2059
 __enumext_standar_unknown_keys:n 3439, 3443, 3447
 __enumext_standar_unknown_keys:nn 3439, 3449, 3451
 \g__enumext_starred_bool 34, 121, 30, 266, 269, 293, 353, 1649, 1927, 1970, 1981, 2008, 2023, 2062, 2168, 2214, 2548, 3091, 4611
 \l__enumext_starred_bool 121, 122, 126, 30, 2586, 2621, 2627, 2675, 2964, 2969, 3680, 4709, 4736, 4971, 4975
 __enumext_starred_columns_set_vii: . . 4388, 4388, 4684
 __enumext_starred_columns_set_viii: . 4388, 4419, 4946
 \l__enumext_starred_first_bool 35, 121, 30, 298, 986, 995, 1892, 2043, 2096, 2103
 __enumext_starred_item_exec: . 94, 3281, 3308, 3350, 3371
 __enumext_starred_item_exec:nn . . 3281, 3281, 3324
 __enumext_starred_item_vii:w . 123, 4767, 4784, 4801
 __enumext_starred_item_vii_aux_i:w . . 4767, 4806, 4809
 __enumext_starred_item_vii_aux_ii:w . 4767, 4807, 4812, 4814
 __enumext_starred_item_vii_aux_iii:w 4767, 4817, 4824
 __enumext_starred_item_viii:w 128, 129, 5027, 5045, 5045
 __enumext_starred_item_viii_aux_i:w . . 129, 5045, 5051, 5054
 __enumext_starred_item_viii_aux_ii:w . 129, 5045, 5052, 5069, 5071
 __enumext_starred_joined_item_vii:n 117, 123, 4450, 4450, 4782
 __enumext_starred_joined_item_viii:n . 117, 128, 4450, 4499, 5025
 __enumext_starred_ref: 45, 782, 818, 3657
 __enumext_starred_ref:n 45, 776, 782, 782
 \g__enumext_starred_series_tl . 57, 1894, 1895, 2064, 2067
 __enumext_starred_unknown_keys:n 3421, 3423, 3425
 __enumext_starred_unknown_keys:nn 3421, 3427, 3429
 __enumext_start_from:NNn 46, 876, 876, 889, 911, 917
 \l__enumext_start_i_int 2006, 2018, 2037
 __enumext_start_item_tmp_vii: 120, 4687, 4767, 4767
 __enumext_start_item_tmp_viii: . . 4949, 5010, 5010
 __enumext_start_item_vii:w 123, 125, 4793, 4798, 4821, 4828, 4876, 4876
 __enumext_start_item_viii:w . . 128, 5037, 5042, 5074, 5149, 5149
 \g__enumext_start_line_tl 35, 30, 286, 301, 358, 2238, 2243, 2248, 2262, 2267, 2272
 __enumext_start_list:nn . 37, 99, 376, 378, 3848, 3981, 4678, 4940
 __enumext_start_list_tag:n . . 3994, 4020, 4886, 5159
 __enumext_start_mini_vii: 121, 4548, 4548, 4728
 __enumext_start_mini_viii: . . 127, 4613, 4613, 4994
 __enumext_start_save_ans_msg: 70, 2080, 2080, 2105
 __enumext_start_store_level: . 102, 3701, 3701, 3847
 __enumext_start_store_level_vii: 122, 4677, 4739, 4739
 \l__enumext_start_vii_int . . . 2011, 2025, 2046
 \l__enumext_start_X_int 96


```

\__enumext_stop_item_tmp_vii: .. 120, 122, 125,
    4686, 4692, 4769, 4878
\__enumext_stop_item_tmp_viii: 128, 4948, 4954,
    5012, 5151
\__enumext_stop_item_vii: 125, 4876, 4878, 4898
\__enumext_stop_item_viii: ... 5149, 5151, 5175
\__enumext_stop_list: 37, 119, 122, 376, 379, 3772,
    3780, 3934, 3941, 4571, 4579, 4636, 4643
\__enumext_stop_list_tag:n ... 3994, 4036, 4901,
    5178
\__enumext_stop_mini_vii: 119, 122, 4548, 4567,
    4732
\__enumext_stop_mini_viii: 127, 4613, 4632, 4998
\__enumext_stop_save_ans_msg: . 70, 2080, 2085,
    3037
\__enumext_stop_start_list_tag: .. 3994, 4028,
    4888, 5161
\__enumext_stop_store_level: .. 103, 104, 3730,
    3730, 3773, 3781
\__enumext_stop_store_level_vii: .. 119, 122,
    4572, 4580, 4739, 4749
\l__enumext_store_active_bool 31, 70, 108, 2035,
    2044, 2112, 2762, 3705, 3718, 3866, 3874, 4247, 4741,
    4751, 4961, 4977
\__enumext_store_active_keys:n . 76, 102, 2369,
    2369, 3698
\__enumext_store_active_keys_vii:n . 76, 121,
    2369, 2379, 4720
\__enumext_store_addto_prop:n 77, 89, 2444, 2444,
    2452, 2612, 3072, 5079
\__enumext_store_addto_seq:n 78, 91, 2453, 2453,
    2457, 2464, 2478, 2486, 2495, 2509, 2517, 2670, 3165
\l__enumext_store_anskey_arg_tl .. 31, 81, 108,
    2618, 2623, 2625, 2630, 2637, 2640, 2650, 2655, 2658,
    2664, 2670
\__enumext_store_anskey_code:n 80, 83, 88, 2609,
    2609, 2755, 3008, 3016
\l__enumext_store_anskey_env_tl .. 31, 87, 108,
    2938, 2942, 2948, 3010, 3018
\l__enumext_store_anskey_opt_tl 31, 87, 88, 108,
    2939, 2966, 2972, 2979, 2985, 2995, 3005, 3014
\__enumext_store_anskey_safe_outer: .... 83
\g__enumext_store_columns_break_bool . 2862,
    2963, 3025
\l__enumext_store_columns_break_bool . 2620,
    2711
\l__enumext_store_current_label_tl 31, 89, 91,
    129, 108, 3053, 3056, 3059, 3065, 3070, 3072, 3129,
    3132, 3135, 3141, 3146, 3156, 3165, 5056, 5061, 5065,
    5078, 5079, 5081
\l__enumext_store_current_label_tmp_tl . 31,
    108
\l__enumext_store_current_opt_arg_tl . 31, 91,
    129, 108, 3175, 3180, 3187, 5067
\__enumext_store_internal_ref: .. 79, 80, 2534,
    2534, 2615
\g__enumext_store_item_join_int .. 2865, 2970,
    2974, 3026
\l__enumext_store_item_join_int .. 2628, 2632,
    2714
\g__enumext_store_item_star_bool . 2867, 2977,
    3027
\l__enumext_store_item_star_bool . 2635, 2716
\g__enumext_store_item_symbol_sep_dim 2872,
    2992, 2997, 3029
\l__enumext_store_item_symbol_sep_dim 2647,
    2652, 2721
\g__enumext_store_item_symbol_tl . 2870, 2983,
    2987, 3028
\l__enumext_store_item_symbol_tl . 2638, 2642,
    2719
\l__enumext_store_keyans_item_opt_sep_
    tl 2306, 2354, 3063, 3067, 3139, 3143, 5059, 5063
\__enumext_store_level_close: . 78, 2458, 2482,
    3734
\__enumext_store_level_close_vii: . 78, 2489,
    2513, 4755
\__enumext_store_level_open: 78, 103, 2458, 2458,
    3713, 3726
\__enumext_store_level_open_vii: .. 78, 2489,
    2489, 4745
\g__enumext_store_name_tl 31, 70, 108, 357, 364,
    365, 366, 367, 2088, 2114, 2237, 2242, 2247, 2261,
    2266, 2271, 3035
\l__enumext_store_name_tl 31, 70, 72, 108, 1921,
    1924, 1948, 1951, 2039, 2048, 2083, 2092, 2093, 2114,
    2115, 2116, 2118, 2119, 2121, 2123, 2124, 2126, 2128,
    2129, 2153, 2446, 2448, 2455, 2598, 2599, 2701, 2944,
    3112, 3113, 3252, 5101
\l__enumext_store_ref_key_bool 80, 2331, 2613,
    2661, 3076, 3153
\l__enumext_store_save_key_vii_bool .. 2381,
    2411
\l__enumext_store_save_key_vii_tl 2383, 2384,
    2412, 2413, 2493, 2501, 2505, 2509
\l__enumext_store_save_key_X_bool .. 76, 126
\l__enumext_store_save_key_X_tl .... 76, 126
\l__enumext_store_upper_level_X_bool .. 126
\__enumext_storing_exec: 70, 85, 2090, 2106, 2110
\__enumext_storing_set:n .. 70, 2075, 2090, 2090
\l__enumext_the_counter_v_tl ..... 862
\l__enumext_the_counter_vii_tl ..... 794
\l__enumext_the_counter_viii_tl ..... 810
\l__enumext_the_counter_X_tl ..... 46
\__enumext_tmp:n 41, 45, 50, 56, 67, 74, 75, 82, 90, 95,
    96, 107, 133, 140, 165, 169, 176, 196, 635, 644, 1828,
    1839, 2071, 2079, 2132, 2150, 2291, 2336, 2337, 2368,
    2387, 2400, 2536, 2543, 2544, 2565, 2578, 2581, 2592,
    3078, 3085, 3399, 3406, 3439, 3446, 3596, 3636, 3637,
    3671
\__enumext_tmp:nn 645, 666, 667, 701, 702, 717, 906,
    931, 1008, 1030, 1031, 1051, 1105, 1113, 1114, 1128,
    1193, 1209, 1210, 1223, 1717, 1733, 3383, 3398
\__enumext_tmp:nnn 718, 734, 735, 736, 737, 764, 780,
    781
\__enumext_tmp:nnnnn 932, 957, 960, 963, 965, 967,
    970, 973
\__enumext_tmp:w ..... 5220, 5223
\l__enumext_tmpa_vii_int 4398, 4401, 4410, 4441
\l__enumext_tmpa_viii_int ..... 4429, 4432
\l__enumext_tmpa_X_dim ..... 176
\l__enumext_tmpa_X_int ..... 176
\l__enumext_topsep_v_skip 1290, 1294, 1493, 4240
\l__enumext_topsep_vii_skip .. 1570, 1579, 1583
\l__enumext_topsep_viii_skip . 1592, 1614, 1618
\__enumext_undefine_anskey_env: . 84, 89, 2795,
    2795, 3046
\__enumext_unskip_unkern: .. 34, 240, 240, 1343,
    1515, 3775, 3776, 3816, 3936, 3937, 3954, 4892, 4893,
    5165, 5166

```

<code>\l__enumext_vspace_a_star_v_bool</code>	1766
<code>\l__enumext_vspace_a_star_vii_bool</code>	1788
<code>\l__enumext_vspace_a_star_viii_bool</code>	1799
<code>\l__enumext_vspace_a_star_X_bool</code>	96
<code>__enumext_vspace_above:</code>	63, 104, 1734, 1734, 3786
<code>__enumext_vspace_above_v:</code>	63, 1762, 1762, 3890
<code>\l__enumext_vspace_above_v_skip</code>	1764, 1768, 1770
<code>__enumext_vspace_above_vii:</code>	64, 121, 1784, 1784, 4725
<code>\l__enumext_vspace_above_vii_skip</code>	1786, 1790, 1792
<code>__enumext_vspace_above_viii:</code>	64, 1784, 1795, 4992
<code>\l__enumext_vspace_above_viii_skip</code>	1797, 1801, 1803
<code>\l__enumext_vspace_b_star_v_bool</code>	1777
<code>\l__enumext_vspace_b_star_vii_bool</code>	1810
<code>\l__enumext_vspace_b_star_viii_bool</code>	1821
<code>\l__enumext_vspace_b_star_X_bool</code>	96
<code>__enumext_vspace_below:</code>	63, 105, 1748, 1748, 3824
<code>__enumext_vspace_below_v:</code>	64, 1773, 1773, 3963
<code>\l__enumext_vspace_below_v_skip</code>	1775, 1779, 1781
<code>__enumext_vspace_below_vii:</code>	64, 122, 1806, 1806, 4735
<code>\l__enumext_vspace_below_vii_skip</code>	1808, 1812, 1814
<code>__enumext_vspace_below_viii:</code>	64, 1806, 1817, 5000
<code>\l__enumext_vspace_below_viii_skip</code>	1819, 1823, 1825
<code>__enumext_widest_from:nNNn</code>	47, 890, 890, 905, 924
<code>\g__enumext_widest_label_tl</code>	29, 42, 63, 623, 627, 631
<code>\l__enumext_wrap_label_opt_v_bool</code>	3469
<code>\l__enumext_wrap_label_opt_vii_bool</code>	123, 4792
<code>\l__enumext_wrap_label_opt_viii_bool</code>	128, 5036
<code>\l__enumext_wrap_label_opt_X_bool</code>	96
<code>\l__enumext_wrap_label_v_bool</code>	3465, 3469, 3476, 3522, 3530, 4290
<code>\l__enumext_wrap_label_vii_bool</code>	123, 4792, 4796, 4804, 4868
<code>\l__enumext_wrap_label_viii_bool</code>	128, 5036, 5040, 5049, 5114, 5122
<code>\l__enumext_wrap_label_X_bool</code>	96
<code>__enumext_wrapper_label_v:n</code>	3528, 3532, 4305
<code>__enumext_wrapper_label_vii:n</code>	4870
<code>__enumext_wrapper_label_viii:n</code>	5120, 5124
<code>\l__enumext_write_aux_file_tl</code>	32, 80, 90, 162, 2601, 2607, 3119, 3125
<code>enumext*</code>	5, 4672
<code>enumXi</code>	586
<code>enumXii</code>	586
<code>enumXiii</code>	586
<code>enumXiv</code>	586
<code>enumXv</code>	586
<code>enumXvi</code>	586
<code>enumXvii</code>	586
<code>enumXviii</code>	586
Environments provide by <code>enumext</code> :	
<code>anskey*</code>	31, 70, 76, 79, 80, 82, 84, 85, 87, 89, 102, 103, 122,

	131, 132, 138, 140
<code>enumext*</code>	28, 29, 32-34, 38-42, 44, 45, 47-51, 53, 60, 61, 64-67, 69-72, 74, 76-84, 87, 89, 90, 95, 96, 101-103, 108, 116, 117, 119, 122, 124-127, 130-133, 135, 139, 142, 143
<code>enumext</code>	28, 29, 33, 34, 38-42, 44-49, 51-56, 58, 61-63, 65-67, 69-72, 74, 76-84, 87, 89, 90, 93-97, 99, 100, 103, 105, 106, 111, 115, 118, 121, 122, 124, 126, 132, 133, 135, 139, 140, 142
<code>keyans*</code>	28, 29, 31-35, 38-41, 44-51, 53, 60, 61, 64, 70, 71, 74, 75, 77, 85, 89, 96, 101, 108, 116, 117, 126, 127, 139, 141, 143
<code>keyanspic</code>	28, 29, 31, 32, 35, 41, 46, 70, 71, 74, 77, 78, 85, 89, 90, 96, 108-114, 141
<code>keyans</code>	28, 29, 31, 32, 34, 35, 38, 39, 41, 42, 46-49, 51, 53, 55, 58, 61-64, 70, 71, 74, 75, 77, 78, 85, 89, 90, 92, 96-100, 106, 108, 110, 111, 114, 118, 127, 139, 141
Environments:	
<code>center</code>	116
<code>description</code>	95, 116
<code>enumerate</code>	116
<code>flushleft</code>	116
<code>flushright</code>	116
<code>itemize</code>	116
<code>list</code>	33, 36, 37, 82, 95, 99, 104, 105, 108, 110-112, 116, 119
<code>lrbox</code>	125
<code>minipage</code>	33, 36, 37, 39, 40, 53, 55-57, 110, 113, 115, 116, 119, 125
<code>multicols</code>	53-57, 61, 103-105
<code>quotation</code>	116
<code>quote</code>	116
<code>scontents</code>	85, 87
<code>tabbing</code>	116
<code>trivlist</code>	116
<code>verbatim</code>	116
<code>verse</code>	116
exp commands:	
<code>\exp_after:wN</code>	5223
<code>\exp_args:Ne</code>	3007, 3015, 3695, 5211
<code>\exp_args:NV</code>	2727, 2882, 3409, 3427, 3449, 5496
<code>\exp_not:N</code>	54, 626, 753, 797, 813, 865, 1061, 1064, 1075, 1076, 1077, 1088, 1089, 1100, 1101, 2666, 2698, 2699, 3158, 3249, 3250, 5098, 5099, 5220
<code>\exp_not:n</code>	288, 303, 316, 324, 332, 692, 712, 753, 797, 813, 865, 1062, 1855, 1864, 2317, 2348, 2430, 2442, 2604, 2632, 2642, 2652, 2666, 2667, 2974, 2987, 2997, 3122, 3160, 3162, 4107, 5325, 5335, 5528, 5533

F

<code>\fbox</code>	2298
<code>\fboxrule</code>	2298
<code>\fboxsep</code>	2298
file commands:	
<code>\file_input_stop:</code>	5932
<code>first</code>	1114
<code>font</code>	645
<code>\footnote</code>	38
<code>\footnote</code>	38, 443, 473
<code>\footnotemark</code>	453, 483
<code>\footnotesize</code>	2699, 3250, 5099
<code>\footnotetext</code>	439
<code>\foreachkeyans</code>	18, 136, 5461

G

<code>\getkeyans</code>	18, 131, 5209
-------------------------	---------------

group commands:

\group_begin: 2697, 2742, 2917, 3004, 3248, 5097, 5266
\group_end: 2704, 2758, 3021, 3255, 5104, 5273

H

\hbadness 4903, 5180
hbox commands:
 \hbox_overlap_left:n 2524, 3314, 4861
 \hbox_set:Nn 615, 4167
 \hbox_set_end: 4902, 5179
 \hbox_set_to_wd:Nnw 4879, 5152
\hfill 675, 680, 686, 687, 1673, 1700, 2666, 3158, 4575, 4639
hook commands:
 \hook_gput_code:nnn 5, 206, 210, 214, 387
 \hook_gremove_code:nn 87, 2933
 \hook_gset_rule:nnnn 388
 \hook_if_empty:nTF 2931
\hyperlink 81, 91
\hyperlink 2666, 3158
\hypertarget 37
\hypertarget 414

I

\IfDocumentMetadataTF . . 505, 521, 535, 548, 3331, 3504,
 4022, 4030, 4038, 4074, 4082, 4090, 4191, 4200, 4208,
 4215, 4220, 4268, 4277, 4363, 4371, 4573, 4637, 4683,
 4691, 4837, 4945, 4953
\IfHyperBoolean 395
\IfPackageLoadedTF 7, 15, 391, 404
\ignorespaces . . 1064, 1077, 1089, 1101, 4180, 4688, 4765,
 4798, 4821, 4828, 4874, 4894, 4950, 5008, 5042, 5074,
 5147, 5167
\inputlineno 288, 303, 316, 324, 332
int commands:
 \int_add:Nn 4483, 4532
 \int_case:nn . . 1238, 1363, 2163, 2189, 2228, 2252
 \int_case:nnTF 242
 \int_compare:nNnTF . . 569, 785, 801, 820, 827, 1333,
 1352, 1506, 1524, 1636, 1655, 1667, 1695, 2276, 2282,
 2766, 2770, 2774, 2782, 2828, 2832, 2836, 3033, 3054,
 3095, 3100, 3105, 3130, 3233, 3677, 3688, 3710, 3723,
 3739, 3754, 3769, 3810, 3875, 3879, 3907, 3932, 3948,
 4139, 4251, 4255, 4453, 4463, 4479, 4502, 4512, 4528,
 4701, 4705, 4743, 4753, 4905, 4917, 4966, 4978, 5182,
 5194, 5379, 5511
 \int_compare_p:nNn . . 257, 267, 279, 280, 294, 295,
 1642, 1643, 2169, 2195, 2549, 2559, 2571, 2572, 2587,
 2628, 2805, 2806, 2817, 2818, 2970, 3720
 \int_decr:N 4482, 4531
 \int_eval:n . . 374, 919, 2448, 2599, 2699, 3113, 3250,
 3611, 3656, 4471, 4520, 5099
 \int_from_alph:n 884, 898
 \int_from_roman:n 886, 900
 \int_gadd:Nn 4484, 4533
 \int_gdecr:N 2172, 2177, 2181, 2185, 2198
 \int_gincr:N 2005, 2010, 2611, 3168, 3268, 3302, 3482,
 3799, 3899, 4295, 4771, 4847, 5014, 5083
 \int_gset:Nn 451, 481, 2221
 \int_gset_eq:NN . . 448, 478, 1904, 1911, 1917, 1923,
 1931, 1938, 1944, 1950
 \int_gzero:N . 345, 346, 347, 1681, 1708, 2288, 3026,
 3815, 3953, 4928, 5206
 \int_if_exist:NnTF 1879, 1915, 1921, 1942, 1948, 2126
 \int_incr:N 2781, 3235, 3676, 3870, 4138, 4700, 4770,
 4965, 5013
 \int_mod:nn 4919, 5196

\int_new:N . 24, 25, 26, 27, 28, 29, 57, 58, 83, 100, 119,
 131, 143, 144, 155, 156, 157, 159, 170, 171, 179, 180,
 181, 182, 183, 1881, 2129
\int_set:Nn 880, 884, 886, 2018, 2025, 2037, 2046, 2918,
 4357, 4358, 4398, 4429, 4452, 4458, 4474, 4501, 4507,
 4523, 4903, 5180, 5375, 5513
\int_set_eq:NN 2006, 2011, 4481, 4530
\int_sign:n 2223
\int_step_function:nnN 2565, 2578, 2592
\int_step_function:nnnN 5517
\int_step_inline:nn 5427
\int_step_inline:nnn 4359
\int_to_roman:n 218, 2545, 2582
\int_use:N 367, 372, 373, 1334, 1353, 1668, 2020, 2027,
 2039, 2048, 3611, 3631, 3656, 3696, 3740, 3749, 3764,
 3770, 4456, 4457, 4469, 4505, 4506, 4518, 5848, 5852,
 5858, 5862
\int_zero:N 3238, 4909, 5186
\item . 93, 97, 122, 125, 127, 130, 380, 2466, 2472, 2497, 2503,
 2625, 3132, 3135, 3320, 3486, 4195, 4196, 4685, 4687,
 4947, 4949, 5081
\item* 5, 15, 74, 3484
item-pos* 3383
item-sym* 3383
\itemindent 100
\itemindent 99
itemindent 1008
\itemsep 4184
\itemwidth . 585, 2298, 3830, 3836, 3967, 3973, 4492, 4496,
 4541, 4545

K

keyans 15, 3976
keyans* 15, 4935
keyanspic 16, 4186
Keys for \anskey provide by enumext:
 break-col 81, 82, 85–87
 item-join 81, 82, 85–87
 item-pos* 81, 82, 85, 86, 88
 item-star 81, 82, 85, 86, 88
 item-sym* 81, 82, 85, 86, 88
Keys for anskey* provide by enumext:
 break-col 81, 82, 85–87
 item-join 81, 82, 85–87
 item-pos* 81, 82, 85, 86, 88
 item-star 81, 82, 85, 86, 88
 item-sym* 81, 82, 85, 86, 88
Keys for environments provide by enumext:
 above* 30, 49, 62–64, 104, 121
 above 30, 49, 62–64, 104, 121, 127
 after 51, 52, 105, 122, 127
 align 30, 43, 92, 93, 95, 98, 124, 138
 base-fix 48, 49, 65, 77, 102
 before* 51, 104, 121, 127
 before 51
 below* 30, 62–64, 105, 122
 below 30, 62–64, 105, 122, 127
 check-ans . 32, 33, 35, 69–71, 73, 74, 77, 88, 91, 105, 106,
 122, 126, 139
 columns-sep 53, 103, 125
 columns 30, 53, 62, 103
 first 51, 52, 125
 font 42, 95, 98, 114, 124
 item-pos* 94, 96
 item-sym* 31, 94, 96

itemindent	30, 49, 50, 93, 94, 97, 98, 125
itemsep	48, 101, 125
label-pos	110, 111, 113, 114
label-sep	110
labelsep	42, 100, 124
labelwidth	41, 42, 44, 46, 47, 100, 124
label	29, 41-43, 46, 47, 111, 115
layout-sep	110
layout-sty	110, 115
layout-top	110
lisparindent	101
list-indent	30, 49, 50, 111
list-offset	49, 50, 105, 107
listparindent	49, 125
mark-ans	74, 75, 77, 82, 92
mark-pos	31, 74, 75, 138
mark-ref	74, 77, 79, 81
mark-sep	31, 74, 75, 77, 92, 129
mini-env	30, 38-40, 53, 61, 62, 77, 104, 116, 118, 119, 121, 122, 127
mini-right*	30, 33, 53, 77, 119, 121, 122
mini-right	30, 33, 53, 61, 77, 119, 121, 122
mini-sep	30, 53, 77, 104
mode-box	42, 93, 95, 98, 99
no-store	32, 69-72, 77, 83, 93, 94
noitemsep	48
nosep	48
parindent	101
parsep	48, 101, 111, 125
partopsep	48
ref	29, 34, 43, 44, 46, 139
resume*	29, 65, 69, 70, 77, 105, 122, 134
resume	29, 36, 65-70, 77, 105, 122, 134
rightmargin	49, 116
save-ans	31, 36, 65-70, 72, 73, 76-78, 83-85, 88, 89, 91, 97, 106, 113, 124, 126, 127, 129, 131, 132, 134, 139
save-key	31, 65, 76, 77, 102, 121
save-pos	77
save-ref	32, 38, 74, 77, 79-81, 90, 91, 98, 129
save-sep	74, 75, 77, 89, 129
series	29, 65-69, 77, 102, 105, 121, 122, 134
show-ans	31, 74, 75, 77, 79, 80, 82, 91, 92, 114, 129
show-length	34, 51, 139
show-pos	31, 74, 75, 79, 80, 82, 91, 114, 129
start*	30, 46, 47, 65
start	30, 33, 46, 47, 65
store-key	76
topsep	48, 49, 111
widest	29, 33, 47
wrap-ans	41, 74, 77, 79, 82
wrap-key	32, 74, 75, 77, 98, 114
wrap-label*	30, 42, 93, 95, 97, 98, 123, 124, 128
wrap-label	30, 42, 93-95, 97, 98, 111, 114, 123, 124, 128
wrap-opt	74, 75, 77, 97, 114
keys commands:	
\keys_define:nn	637, 647, 669, 704, 720, 766, 835, 908, 934, 976, 1010, 1033, 1107, 1116, 1195, 1212, 1719, 1830, 2073, 2134, 2293, 2339, 2389, 2394, 2709, 2860, 2896, 3385, 3401, 3421, 3441, 4096, 5237, 5337, 5453, 5461
\keys_if_exist_p:nn	5449, 5450
\l_keys_key_str	83, 86, 2727, 2882, 3409, 3427, 3449, 5496, 5604
\keys_precompile:nnN	132, 202, 202, 5239, 5243, 5247, 5251, 5255, 5259, 5479

\keys_set:nn	661, 1002, 1218, 1724, 1729, 1967, 1972, 2059, 2067, 2747, 3690, 3695, 3886, 4114, 4116, 4118, 4120, 4122, 4124, 4126, 4128, 4130, 4132, 4149, 4718, 4987, 5341, 5346, 5347, 5348, 5349, 5352, 5357, 5358, 5359, 5360, 5361, 5362, 5363, 5395, 5505
\keys_set_known:nn	3014
keyval commands:	
\keyval_parse:NNn	1844, 2419, 5313

L

label	718, 764, 835
label-pos	4096
label-sep	4096
Labels provide by enumext:	
\Alph*	41, 42
\Roman*	41, 42
\alph*	41, 42
\arabic*	34, 41, 42
\roman*	41, 42
labelsep	645
\labelwidth	42
labelwidth	645
\lastnodetype	242
layout-sep	4096
layout-sty	4096
layout-top	4096
\leftmargin	100
\leftmargin	99, 4179
legacy commands:	
\legacy_if:nTF	4832, 4835, 5130, 5133
\legacy_if_gset_false:n	575, 4588
\legacy_if_set_false:n	4834, 5132
\legacy_if_set_true:n	4797, 4820, 4827, 4841, 5041, 5073
\linewidth	104
\linewidth	3794, 3830, 3896, 3967, 4356, 4401, 4432, 4554, 4619
\list	378
list-indent	1008
list-offset	1008
\listparindent	4182
listparindent	1008

M

\makebox	115
\makebox	2526, 3367, 3554, 4285, 4865, 5142
\makelabel	93, 95, 98, 115
\makelabel	93, 97, 3347, 3363, 3538, 3550
mark-ans	2291, 2337, 4096
mark-pos	2291, 2337, 4096
mark-ref	2291
mark-sep	2291, 2337, 4096
mini-env	1193
mini-sep	1193
\minipage	384
\miniright	11, 61, 1634, 1685, 1712, 3813, 3951
mode commands:	
\mode_if_math:TF	2790, 2844
\mode_if_vertical:TF	1272, 1300, 1320, 1344, 1495, 1516
\mode_leave_vertical:	991, 998, 1061, 1075, 2522, 3312, 4859
mode-box	635

msg commands:

`\msg_error:nn` . . . 1687, 1714, 2751, 2784, 2788, 2842, 2950, 3877, 3881, 4141, 4198, 4253, 4703, 4968, 4980, 5364, 5423

`\msg_error:nnn` 743, 789, 805, 857, 1638, 1645, 1652, 1683, 1710, 1979, 1983, 2098, 2733, 2792, 2810, 2822, 2830, 2834, 2838, 2846, 2888, 3415, 3433, 3455, 4707, 4973, 5225, 5234, 5306, 5411, 5442, 5451, 5488, 5509

`\msg_error:nnnn` 2736, 2764, 2768, 2772, 2776, 2891, 3418, 3436, 3458, 3868, 4249, 4257, 4963, 5285, 5491

`\msg_error:nnnnn` 691, 711, 2316, 2347, 4106

`\msg_fatal:nn` 3678

`\msg_fatal:nnn` 589

`\msg_info:nnn` 9, 12, 17, 20, 393, 406

`\msg_line_context:` . . . 5569, 5574, 5579, 5608, 5613, 5618, 5633, 5648, 5652, 5656, 5660, 5664, 5668, 5675, 5682, 5688, 5702, 5706, 5711, 5715, 5719, 5723, 5728, 5732, 5736, 5740, 5745, 5780, 5784, 5789, 5794, 5798, 5803, 5879, 5883, 5888, 5893, 5898, 5902, 5906, 5910, 5914, 5918, 5922, 5926, 5930

`\msg_log:nnn` 2118, 2123, 2128

`\msg_log:nnnnn` 371, 2261, 2266, 2271

`\msg_log:nnnnnn` 363

`\msg_new:nnn` 5536, 5540, 5544, 5548, 5553, 5566, 5571, 5576, 5581, 5590, 5598, 5602, 5606, 5611, 5616, 5631, 5646, 5650, 5654, 5658, 5662, 5666, 5670, 5679, 5685, 5691, 5695, 5699, 5704, 5709, 5713, 5717, 5721, 5726, 5730, 5734, 5738, 5743, 5778, 5782, 5787, 5792, 5796, 5801, 5877, 5881, 5886, 5891, 5896, 5900, 5904, 5908, 5912, 5916, 5920, 5924, 5928

`\msg_new:nnnn` . . . 5557, 5748, 5757, 5766, 5772, 5805, 5815, 5825, 5835, 5845, 5855, 5865, 5871

`\msg_term:nnnn` . . . 2082, 2087, 3620, 3630, 3662, 3667

`\msg_term:nnnnn` 2242

`\msg_warning:nn` 3812, 3950

`\msg_warning:nnnn` 2279, 2285, 3568, 3573, 4455, 4468, 4504, 4517

`\msg_warning:nnnnn` 2237, 2247

`\multicolsep` 103

`\multicolsep` 1337, 1509, 3760, 3923

N

`\NeedsTeXFormat` 3

`\NewCommandCopy` 380

`\newcounter` 592

`\NewDocumentCommand` 1634, 2739, 4245, 5209, 5264, 5371, 5420, 5498

`\NewDocumentEnvironment` . . 3842, 3976, 4186, 4672, 4935

`\newenvsc` 2853

`\newlabel` 38

`\newlabel` 426

`no-store` 2132

`\noindent` 3801, 4563, 4628, 4908, 5185

`\nointerlineskip` 1346, 1349, 1518, 1521, 1675, 1702, 4563, 4628

`noitemsep` 932

`\nopagebreak` 1283, 1311, 1346, 1349, 1518, 1521, 1625, 1631

`\normalfont` 2698, 3249, 5098

`nosep` 932

P

Packages:

`caption` 119

`enumext` 28, 41, 43, 69, 95, 100, 110, 138

`enumitem` 41

`expl3` 115

`footnotehyper` 37, 39, 40

`hyperref` 32, 33, 37, 38, 81, 91, 124, 137

`latex-lab-block` 37

`ltxcmd` 37

`ltsockets` 108

`lua-visual-debug` 56

`multicol` 28, 137

`scontents` 28, 84, 85

`shortlst` 115, 120, 125

`tagpdf` 108

`\par` . . . 1283, 1311, 1349, 1521, 1625, 1631, 1670, 1675, 1697, 1702, 2674, 3777, 3938, 3956, 4231, 4234, 4376, 4590, 4605, 4651, 4665, 4908, 5185

para commands:

`\para_end:` 4925, 5203

`\parbox` 2298

`\parindent` 4890, 5163

`\parsep` 54, 111

`\parsep` 992, 3653, 4163, 4172

`parsep` 932

`\parskip` 4891, 5164

`\partopsep` 3654, 3954, 4183

`partopsep` 932

peek commands:

`\peek_meaning:NTF` 4776, 4790, 4805, 4816, 5019, 5034, 5050

`\peek_meaning_remove:NTF` 4783, 5026

`\peek_remove_spaces:n` 3491

`\phantomsection` 37

`\phantomsection` 415

prg commands:

`\prg_do_nothing:` 419

`\prg_new_protected_conditional:Npnn` . . . 220

`\prg_replicate:nn` 237

`\prg_return_false:` 224

`\prg_return_true:` 223

`\printkeyans` 18, 132, 5264

prop commands:

`\prop_const_from_keyval:Nn` 5412

`\prop_count:N` 365, 2448, 2599, 2701, 3113, 3252, 5101, 5514

`\prop_get:NnNTF` 5438

`\prop_gput_if_not_in:Nnn` 2446

`\prop_if_exist:NTF` 2116, 5229, 5507

`\prop_item:Nn` 5231, 5531

`\prop_new:N` 2119

`\ProvidesExplPackage` 4

R

`\raggedcolumns` 3763, 3926

`\raisebox` 4318

`\ref` 79, 89

`ref` 718, 764, 835

`\refstepcounter` 4844, 5135

regex commands:

`\regex_match:nnTF` . . . 222, 883, 885, 897, 899, 2946

`\regex_replace_once:nnN` 230

`\renewcommand` 753, 797, 813, 865

`\RenewDocumentCommand` . 443, 473, 1685, 1712, 3320, 3347, 3363, 3486, 3538, 3550, 4196

`\RequirePackage` 13, 21

`resume` 1828

`resume*` 1828

`rightmargin` 1008

`\Roman` 42, 46, 47

\Roman	611	\skip_sub:Nn	1376, 1378, 1392, 1394, 1416, 1418, 1432, 1434, 1455, 1457, 1471, 1473, 1544, 1545, 1556, 1557
\roman	42, 46, 47	\skip_use:N	1228, 1232, 1276, 1280, 1284, 1304, 1308, 1318, 1324, 1737, 1741, 1744, 1751, 1755, 1758, 3777
\roman	612, 736, 5254	\skip_vertical:N	576, 579, 1000, 4589, 4603, 4927, 5205
S			
\s	2947	\skip_vertical:n	999, 4926, 5204
save-ans	2071	\skip_zero:N	1336, 1350, 1488, 1489, 1490, 1508, 1522, 3654, 3760, 3923, 4183, 4184
save-key	2387	\skip_zero_new:N	1567, 1589, 1590, 1591
save-ref	2291	\c_zero_skip	576, 579, 1000, 1241, 1250, 1259, 1407, 1446, 1570, 1592, 1737, 1751, 1764, 1775, 1786, 1797, 1808, 1819, 4589, 4603, 4927, 5205
save-sep	2291, 2337, 4096	\small	5242, 5246, 5250, 5254, 5258, 5262
scan commands:			
\scan_stop:	4195, 4685, 4947, 5220, 5223	\smash	3365, 3552
scontents internal commands:			
\l_scontents_fname_out_tl	2906	socket commands:	
_scontents_parse_environment_keys:n	2912	\socket_assign_plug:nn	4024, 4032, 4040, 4076, 4084, 4092
_scontents_rescan_tokens:n	2919	\socket_new:nn	3994, 4044
\l_scontents_storing_bool	2904	\socket_new_plug:nnn	3995, 4003, 4011, 4045, 4053, 4062
\l_scontents_writing_bool	2905	\socket_use:n	4077, 4085, 4093
seq commands:			
\seq_clear:N	5373, 5516	\socket_use:nn	4025, 4033, 4041
\seq_const_from_clist:Nn	5366	start	906
\seq_count:N	366, 4382, 5377	start*	906
\seq_gclear:N	468, 469, 498, 499	start-list-tags	3994, 4044
\seq_gput_right:Nn	454, 455, 484, 485, 2455	\stepcounter	447, 477, 4166, 4311
\seq_if_empty:NTF	461, 491, 5279, 5391	stop-list-tags	3994, 4044
\seq_if_exist:NTF	2121, 5277	stop-start-tags	3994, 4044
\seq_if_in:NnTF	5283	str commands:	
\seq_item:Nn	2944, 4369	\c_backslash_str	2792, 5569, 5574, 5579, 5584, 5586, 5588, 5593, 5595, 5693, 5697, 5701, 5711, 5715, 5723, 5724, 5728, 5740, 5741, 5745, 5746, 5767, 5769, 5773, 5775, 5803, 5866, 5868, 5872, 5874, 5883, 5884, 5888, 5893, 5894, 5898, 5902, 5906
\seq_map_function:NN	5382	\c_colon_str	2598, 3112, 5220
\seq_map_inline:Nn	5292, 5300, 5392, 5393	\c_left_brace_str	5674, 5681, 5687
\seq_map_pairwise_function:NNN	463, 493	\c_right_brace_str	5674, 5681, 5687
\seq_new:N	120, 121, 123, 141, 172, 173, 174, 175, 2124	\str_case:nn	250, 309, 3195
\seq_pop_left:NN	5381	\str_case:nnTF	1851, 1859, 2426, 2434, 5320, 5329
\seq_put_right:Nn	4259, 5389, 5405, 5526	\str_clear:N	3687, 4717
\seq_set_from_clist:Nn	5374	\str_count:n	237
\seq_set_map_e:NNn	5383	\str_if_empty:NTF	1868, 1909, 1936
\seq_use:Nn	202, 203, 5522	\str_if_eq:nnTF	3612, 3658, 5422
series	1828	\str_if_in:nnTF	5216
\setcounter	894, 898, 900, 3611, 3656, 4228	\str_new:N	80, 128, 146, 189
\setenumext	6, 133, 5371	\str_set:Nn	676, 682, 688, 707, 708, 709, 2313, 2314, 2344, 2345, 4101, 4104
\setenumextmeta	6, 135, 5412	\str_use:N	3369
show-ans	2291, 2337, 4096	\strut	3365, 3552
show-length	1105	\strutbox	1355, 1358, 1369, 1370, 1381, 1383, 1398, 1401, 1409, 1410, 1421, 1423, 1438, 1441, 1448, 1449, 1460, 1462, 1477, 1480, 1526, 1529, 1537, 1538, 1546, 1547, 1559, 1561, 1572, 1573, 1576, 1583, 1596, 1604, 1610, 1618, 4175, 4181, 4231, 4239, 4324
show-pos	2337, 4096	T	
skip commands:			
\skip_add:Nn	1243, 1252, 1261, 1274, 1278, 1302, 1306, 1322, 1380, 1382, 1396, 1399, 1420, 1422, 1436, 1439, 1459, 1461, 1475, 1478, 1497, 1546, 1547, 1558, 1560, 4172, 4181	tag commands:	
\skip_gset:Nn	1573, 1577, 1581	\tag_mc_begin:n	4001, 4051, 4060
\skip_gzero_new:N	1568, 1569	\tag_mc_begin_pop:n	4017, 4069, 4223, 4225
\skip_horizontal:N	1076, 1088, 1100, 4862, 4874, 4912, 5147, 5189	\tag_mc_end:	4005, 4055, 4064
\skip_horizontal:n	1062, 2523, 2531, 3313, 3315, 4761, 4860, 4894, 5004, 5167	\tag_mc_end_push:	3998, 4048, 4211
\skip_if_eq:nnTF	1241, 1250, 1259, 1366, 1406, 1446, 1534, 1570, 1592, 1736, 1750, 1764, 1775, 1786, 1797, 1808, 1819	\tag_resume:n	3997, 4047, 4202, 4210, 4279, 4373, 4573, 4637
\skip_new:N	77, 78, 79, 84, 85, 86, 87, 88, 89, 194		
\skip_set:Nn	1226, 1230, 1288, 1292, 1316, 1369, 1370, 1388, 1409, 1410, 1428, 1448, 1449, 1467, 1491, 1537, 1538, 1552, 1572, 1576, 1594, 1598, 1602, 1608, 1612, 1616, 4156		
\skip_set_eq:NN	1327, 1328, 1330, 1337, 1502, 1503, 1504, 1509, 3609, 3652, 3653, 4891, 5164		

\tag_struct_begin:n	3999, 4000, 4007, 4008, 4009, 4049, 4050, 4057, 4058, 4059, 4212
\tag_struct_end:n	4006, 4013, 4014, 4015, 4016, 4056, 4065, 4066, 4067, 4068, 4222, 4224, 4691, 4953
\tag_suspend:n	4018, 4070, 4193, 4204, 4217, 4270, 4365, 4683, 4945
\tag_tool:n	4203
TeX and L ^A T _E X 2 _ε commands:	
\@auxout	424
\@currenvir	250, 309
\protected@write	424
tex commands:	
\tex_newlinechar:D	2918
text commands:	
\text_expand:n	5212
\textasteriskcentered	2310, 2342
\textborn	3389
\textreferencemark	2329
\thepage	430
tl commands:	
\c_space_tl	3187, 4300, 5618, 5633, 5656, 5660, 5847, 5848, 5857, 5858, 5918, 5922
\tl_clear:N	674, 681, 2289, 2373, 2383, 2404, 2412, 2618, 2938, 2939, 3053, 3129, 5056
\tl_clear_new:N	621
\tl_const:Nn	46, 605
\tl_gclear:N	357, 358, 359, 1889, 1894, 3028, 3358, 3378, 4609, 4669, 4863
\tl_gclear_new:N	1876
\tl_gput_right:Nn	606
\tl_greplace_all:Nnn	627
\tl_gset:Nn	285, 286, 300, 301, 1877, 1890, 1895, 2114, 2942, 3289, 4811
\tl_gset_eq:NN	623, 3285, 4856
\tl_if_blank:nTF	2731, 2749, 2886, 3413, 3431, 3453, 4854, 5486
\tl_if_empty:nTF	741, 759, 787, 803, 822, 829, 855, 871, 1902, 1907, 1929, 1934, 1992, 2056, 2064, 2093, 2153, 2462, 2493, 2638, 2983, 3005, 3035, 3063, 3139, 3180, 3310, 4380, 5059, 5403
\tl_if_empty:nTF	1957
\tl_if_exist:nTF	1962
\tl_if_novalue:nTF	445, 475, 2745, 3061, 3137, 3173, 3264, 3283, 3291, 3463, 3685, 4147, 4715, 4985, 5057
\tl_map_inline:Nn	228, 624
\tl_new:N	38, 39, 40, 43, 48, 49, 52, 53, 59, 61, 62, 64, 65, 101, 102, 103, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 122, 124, 125, 126, 132, 135, 136, 153, 162, 163, 164, 167, 188
\tl_put_left::Ne	2972
\tl_put_left:Nn	2470, 2501, 2623, 2966, 2979, 2985, 2995, 4593, 4654, 5078, 5081
\tl_put_right:Nn	622, 751, 795, 811, 863, 2474, 2505, 2552, 2562, 2575, 2590, 2596, 2601, 2625, 2630, 2637, 2640, 2650, 2655, 2658, 2664, 3056, 3059, 3065, 3070,

3097, 3102, 3107, 3110, 3119, 3132, 3135, 3141, 3146, 3156, 5061, 5065	
\tl_remove_all:Nn	5402
\tl_remove_once:Nn	2540, 3082
\tl_replace_all:Nnn	626, 5437
\tl_reverse:N	2539, 2541, 3081, 3083
\tl_set:Nn	54, 254, 264, 313, 314, 321, 322, 329, 330, 591, 675, 680, 686, 687, 740, 784, 854, 1059, 1073, 1086, 1098, 1991, 2092, 2374, 2384, 2405, 2413, 2695, 2906, 3175, 3246, 5067, 5095, 5400, 5436, 5506
\tl_set_eq:NN	632, 746, 749, 792, 794, 808, 810, 860, 862, 2538, 3080, 3093
\tl_to_str:n	1962, 1968, 1973, 5212
\tl_trim_spaces:n	622, 5389, 5400, 5406, 5422
\tl_use:N	628, 631, 761, 824, 831, 873, 1131, 1135, 1139, 1143, 1147, 1151, 1155, 1159, 1163, 1167, 1171, 1175, 1179, 1183, 1187, 1191, 2528, 2545, 2553, 2564, 2577, 2582, 2593, 3272, 3278, 3306, 3349, 3351, 3357, 3372, 3466, 3470, 3477, 3540, 3543, 3545, 3558, 3849, 3982, 4298, 4304, 4600, 4661, 4867, 4895, 4896, 5144, 5168, 5173, 5267, 5268, 5269, 5270, 5271, 5288, 5385, 5504
token commands:	
\token_to_str:N	426
\topsep	3954, 4181
topsep	932
\topskip	1336, 1508
U	
\u	231, 2947
\unkern	245
unknown	3399, 3421, 3439
\unskip	244
use commands:	
\use:N	238, 3354, 3375, 3851
\use:n	1842, 2417, 5218, 5311
\use_none:nn	418, 5443
\usecounter	3610, 3655
V	
\value	1905, 1911, 1918, 1924, 1932, 1938, 1945, 1951
vbox commands:	
\vbox_set:Nn	4272
\vbox_set_top:Nn	4598, 4659
\vspace	992, 1741, 1744, 1755, 1758, 1768, 1770, 1779, 1781, 1790, 1792, 1801, 1803, 1812, 1814, 1823, 1825
W	
widest	906
wrap-ans	2291
wrap-key	2291, 2337, 4096
wrap-label	645
wrap-label*	645
wrap-opt	2291, 2337, 4096
Z	
\z	2947