

enumext

ENUMERATE EXERCISE SHEETS

v1.6 2025-06-21^{*}

©2024–2025 by Pablo González L[†]

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* package.

Contents

1	Introduction	1	6	The storage system	11
1.1	Description and usage	2	6.1	Keys for storage system	12
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 marks	12
1.3.1	Internal counters	3	6.1.3	Keys for debug and checking	14
1.3.2	Public dimension	3	6.2	The command <code>\anskey</code>	14
1.3.3	Support for multicol	4	6.2.1	Keys for <code>\anskey</code>	14
1.3.4	Support for minipage	4	6.3	The environment <code>anskey*</code>	15
1.3.5	The <code>\label</code> and <code>\ref</code> system	4	6.3.1	Keys for <code>anskey*</code>	15
1.3.6	Support for <code>\footnote</code>	4	6.4	The environment <code>keyans</code>	16
2	The environments provided	5	6.4.1	The <code>\item*</code> in <code>keyans</code>	16
2.1	The environment <code>enumext</code>	5	6.5	The environment <code>keyanspic</code>	17
2.2	The environment <code>enumext*</code>	5	6.5.1	Keys for <code>keyanspic</code>	17
2.3	The command <code>\item*</code>	5	6.5.2	The command <code>\anspic</code>	17
2.3.1	Keys for <code>\item*</code>	6	6.6	Printing stored content	19
2.4	The command <code>\item</code> in <code>enumext*</code>	6	6.6.1	The command <code>\getkeyans</code>	19
3	The command <code>\setenumext</code>	6	6.6.2	The command <code>\foreachkeyans</code>	19
4	The command <code>\setenumextmeta</code>	6	6.6.3	The command <code>\printkeyans</code>	19
5	The keyval system	7	7	Full examples	21
5.1	Keys for label and ref	7	8	Tagged PDF examples	23
5.2	Keys for penalties	8	9	The way of non-enumerated lists	24
5.3	Keys for spaces	8	10	References	26
5.3.1	Vertical spaces	8	11	Change history	27
5.3.2	Horizontal spaces	9	12	Index of Documentation	28
5.4	Keys for add code	10	13	Implementation	30
5.5	Keys for start, series and resume	10	14	Index of Implementation	151
5.6	Keys for multicol	11			
5.7	Keys for minipage	11			
5.7.1	The command <code>\miniright</code>	11			
5.7.2	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 L^AT_EX team for their great work and to the different members of the TeX-SX community who have provided great answers and ideas. Here a note of the main ones:

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

^{*}This file describes a documentation for v1.6, last revised 2025-06-21.

[†]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] package, need to have a modern T_EX distribution such as T_EX Live or MiK_TTeX. 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 2025-06-01.

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

(a) $\alpha > \delta$

(b) L^AT_EXze is cool?

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

(b) Usually uses the package manager?

(c) Rate the following package and class

i. xsim-exam

ii. xsim

iii. exsheets

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

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

*

$(x - 1)^2$

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

*

$3(x + y + z)$

3. True False

(a) $\alpha > \delta$

*

False

(b) L^AT_EXze is cool?

*

Very True!

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

*

Yes

(b) Usually uses the package manager?

*

Yes, dnf

(c) Rate the following package and class

i. xsim-exam

*

doesn’t exist for now :(

ii. xsim

*

very good

iii. exsheets

*

obsolete
- Or we are interested in referring to a specific question and its “answer”, for example:
- The answer to 3.(b) is “Very True!” and the answer to 4.(c).ii is “very good”.
- Or we are interested in printing all the “answers”:
1. $(x - 1)^2$

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

3. (a) False

(b) Very True!

4. (a) Yes

⌘

(b) Yes, dnf

⌘

(c) i. doesn’t exist for now :(

⌘

ii. very good

⌘

iii. obsolete

⌘
- Another very common thing to use in my work is “multiple choice questions”, for example:
1. First type of questions

A) value

B) correct

C) value

D) value

2. Second type of questions

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

II. $\alpha = \delta$

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

A) I only

B) II only

C) I and II only

D) I and III only

E) I, II, and III

★ 3. Third type of questions

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

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

A) value

B) value

C) value

D) value

E) value

4. Question with image and label below:

A

A)

B


B)

A

C)

A

D)



E)

5. Question with image on right side:

A) value

B) value

C) value

D) correct

E) value

B

©2024–2025 by Pablo González L

2 / 167

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.

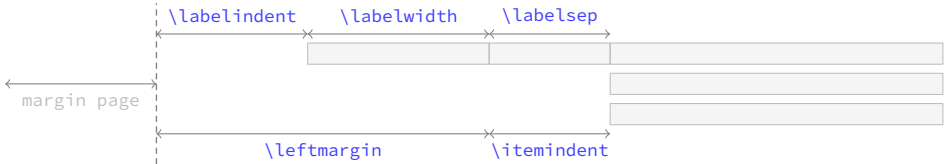


Figure 1: Representation of horizontal lengths in `enumitem`.

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.

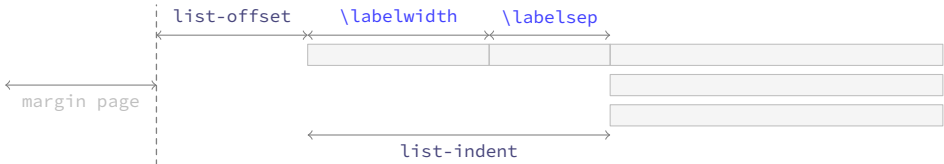


Figure 2: Representation of horizontal lengths concept in `enumext`.

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.

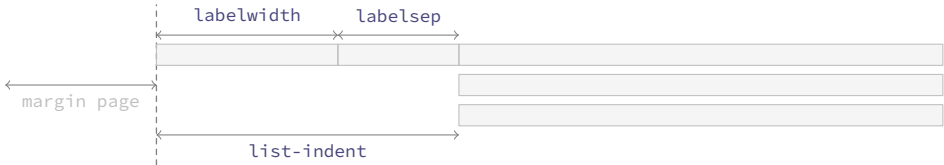


Figure 3: Default horizontal lengths `list-offset=0pt`, `list-indent=\labelwidth+\labelsep` in `enumext`.

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.

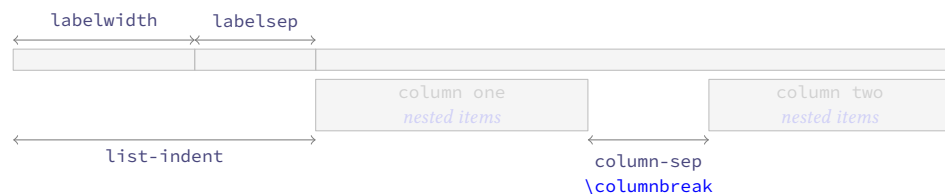


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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 §5.6).

- 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 §5.7).

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:

```
\IfDocumentMetadataF
{
  \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 \LaTeX , `\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`[16] and `tasks`[17] 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 `\makebox` are redefined using `\makebox` by default. If `enumext` or `keyans` is used in the `enumext*` environment the key must be activated manually.

`label` = { $\langle \backslash\alpha^* | \backslash\Alpha^* | \backslash\arabic^* | \backslash\roman^* | \backslash\Roman^* \rangle$ } default: *by levels*

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

- 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{\langle \backslash\alpha^* \rangle}` 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 the `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 set by `label` key using ‘0’ for `\arabic*`, ‘M’ for `\Alpha*`, ‘m’ for `\alpha*`, ‘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 `\Alpha`, `\alpha`, `\Roman` or `\roman` to a *value* for the current counter defined by `label` key, then calculating the *width* by means of a box. For example `widest={XXIII}` or `widest={23}` are equivalent. This key is useful when the default values of the `labelwidth` key are smaller than those actually used.

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

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

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

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

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

Wraps the *current* $\langle label \rangle$ defined by `label` key referenced by $\{ \#1 \}$ 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* ‘ $\{ \# \#1 \}$ ’. 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*` = { $\langle code \{ \#1 \} more code \rangle$ } 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 penalties

Page breaks in the provided environments are controlled by the following three parameters, which work together to ensure they look good, avoiding unsightly page breaks that could distort the output.

`beginpenalty = {\integer}` default: *-51*

Set the *page breaking* penalty for breaking at the beginning of the `enumext`, `enumext*`, `keyans`, and `keyans*` environments. Internally sets the value of `\@beginparpenalty`.

`midpenalty = {\integer}` default: *-51*

Set the *page breaking* penalty for breaking between items of the `enumext`, `enumext*`, `keyans`, and `keyans*` environments. Internally sets the value of `\@itempenalty`.

`endpenalty = {\integer}` default: *-51*

Set the *page breaking* penalty for breaking at the end of the `enumext`, `enumext*`, `keyans`, and `keyans*` environments. Internally sets the value of `\@endparpenalty`.

- The values passed to these *keys* affect the nested environments in which they were set and cannot be reset. L^AT_EX default is `-\@lowpenalty`, that is, `-51`. Because it is negative, it somewhat encourages a page break at each spot. Change it with, e.g., `\@beginparpenalty=9999`; a value of `10000` prohibits a page break. Please, refer to your L^AT_EX or T_EX manual about how penalties control page breaks.

5.3 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`.

5.3.1 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` $\langle \text{value forbidden} \rangle$ 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` $\langle \text{value forbidden} \rangle$ 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` $\langle \text{value forbidden} \rangle$ 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.

Extra vertical spaces

- The following $\langle \text{keys} \rangle$ 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 ‘*’ $\langle \text{keys} \rangle$ applies `\vspace*` so that \LaTeX does *not discard* this space at page break.

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

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

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

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

5.3.2 Horizontal spaces

`list-offset` = $\{ \langle \text{rigid length} \rangle \}$ 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` = $\{ \langle \text{rigid length} \rangle \}$ 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 $\langle \text{label} \rangle$ 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 `labelwidth` plus `labelsep`. Passing a value to this key is equivalent to setting the value for the `list-offset` key.

`itemindent` = $\{ \langle \text{rigid length} \rangle \}$ 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 = {⟨rigid length⟩}` 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 = {⟨rigid length⟩}` 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.

5.4 Keys for add code

The following *⟨keys⟩* should be used with “caution”, they are intended to inject `{⟨code⟩}` 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 \TeX which is defined (simplified) as plain form `\list{⟨arg one⟩}{⟨arg two⟩}`. Using the `before*` key does not allow access to the `list` parameters defined by `[⟨key = val⟩]`.

`before = {⟨code⟩}` default: *not used*

Execute `{⟨code⟩}` “before” the environment starts. The `{⟨code⟩}` must be passed between braces, is executed “after” all calculations related to the *list parameters* in the environment and the *⟨keys⟩* sets by `[⟨key = val⟩]` have been performed, with the exception of the *⟨keys⟩* `start` and `start*`, that is, in the *second argument* of the list: `\begin{list}{⟨arg one⟩}{⟨arg two⟩}{⟨code⟩}`.

`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 the *⟨keys⟩* sets in `[⟨key = val⟩]` of the environment that is, “before” the arguments defining the list 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`, `keyans*` and `keyanspic` environments. 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.

5.5 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`.

- For compatibility with tagged PDF, the *start counter* are set “after” the *second argument* to the `list` environment and “before” the execution of the first `\item` and the `first` key: `\begin{list}{⟨arg one⟩}{⟨arg two⟩}\setcounter{enumX}\item`.
- 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 $\{\langle series\ name\rangle\}$ the keys `series`, `resume`, `resume*`, `save-ans`, `save-key`, `start*` and `start`. When using the key `resume=\{\langle series\ name\rangle\}` it will have hierarchy in the $\langle keys\rangle$ that are saved in $\{\langle series\ name\rangle\}$, in order to establish the value of a $\langle key\rangle$ already saved in $\{\langle series\ name\rangle\}$ it must be placed to the “right” of `resume=\{\langle series\ name\rangle\}`, 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*.

5.6 Keys for multicols

`columns = \{\langle integer\rangle\}`

default: 1

Set the *number of columns* to be used by the `multicols` environment within the environments `enumext` and `keyans`. The value must be a positive integer less than or equal to 10. In the `enumext*` and `keyans*` environments they correspond to the default number of columns (without joining) and internally adjust the value of `\itemwidth`.

`columns-sep = \{\langle rigid\ length\rangle\}`

default: by level

Set the *space between columns* used by the `multicols` environment within the environments `enumext` and `keyans`. 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. In the `enumext*` and `keyans*` environments they correspond to the *space between columns* (without joining) and internally adjust the value of `\itemwidth`.

5.7 Keys for minipage

`mini-env = \{\langle rigid\ length\rangle\}`

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 = \{\langle rigid\ length\rangle\}`

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`.

5.7.1 The command `\miniright`

```
\miniright \begin{enumext}[mini-env=\{\langle rigid\ length\rangle\}] \item's before \item \miniright \langle content\rangle \end{enumext}
\begin{enumext}[mini-env=\{\langle rigid\ length\rangle\}] \item's before \item \miniright* \langle content\rangle \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”.

5.7.2 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 = \{\langle content\rangle\}`

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 $\{\langle content\rangle\}$ must be passed between braces.

`mini-right* = \{\langle content\rangle\}`

default: not used

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

6 The storage system

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

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

By executing the key `save-ans=\{\langle store\ name\rangle\}` 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

$\langle content \rangle$ passed to $\backslash anskey$ or $anskey^*$, the current $\langle labels \rangle$ for $\backslash item^*$ and $\backslash anspic^*$ in the environments $keyans$, $keyans^*$ and $keyanspic$ will be “stored” in a *sequence* $\{\langle store name \rangle\}$ and at the same time will be “stored” (without the “structure” or *optional argument*) in a *prop list* $\{\langle store name \rangle\}$.

For security reasons the *optional argument* of the inner levels or the nested environment are *filtered* by excluding all $\langle keys \rangle$ 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* $\{\langle store name \rangle\}$ set by `save-ans` key.

6.1 Keys for storage system

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

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

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

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

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

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

The “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 $\backslash item^*$ and $\backslash anspic^*$ 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 $\backslash ref\{\langle store name : position \rangle\}$, 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 $\backslash ref\{test:4\}$ 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 marks

The `enumext` package provides a set of $\langle keys \rangle$ to set and manipulate “symbol marks” associated with “answers” and how they are displayed and stored in the *sequence* and *prop list*.

The $\langle keys \rangle$ available for the $\backslash anskey$ command and the $anskey^*$ environment can be passed “only” in the *optional argument* in the “first level” of the `enumext` or `enumext^*` environment.

The $\langle keys \rangle$ available for the $keyans$ and $keyans^*$ environments can be passed locally in the *optional argument*, at the “first level” of the `enumext` or `enumext^*` environment or via the `\setenumext` command with one

minor difference, when $\langle keys \rangle$ are passed through the “first level” of the `enumext` or `enumext*` environment they are set in “both” environments, but when they are passed using the `\setenumext` command they are set “individually” in each environment.

`show-ans = { $\langle true | false \rangle$ }` default: `false`

Display the *symbol* set by the `mark-ans` key to the left of the *mandatory argument* $\langle content \rangle$ passed to the `\anskey` command and $\langle body \rangle$ for the `anskey*` environment using the `wrap-ans` key if set.

For `\item*` and `\anspic*` the `keyans`, `keyans*` and `keyanspic` environments it will display the *symbol* set by the `mark-ans*` key to the left of the current $\langle label \rangle$ and *optional argument*. If the *optional argument* is present in `\item*` or `\anspic*` it will be shown using `wrap-opt` key.

Keys for `\anskey` and `anskey*`

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

Sets the *symbol* to be displayed in the left margin for `\anskey` command and `anskey*` environment 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 = { $\langle left | right | center \rangle$ }` default: `left`

Sets the *aligned* of the “*symbol*” defined by `mark-ans` key for `\anskey` command and `anskey*` environment. 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 *mandatory argument* $\langle content \rangle$ passed to the `\anskey` command and the *body* in `anskey*` environment.

`wrap-ans = { $\langle code \{ \#1 \} \text{ more code} \rangle$ }` default: `\fbox+\parbox{\#1}`

Wraps the *mandatory argument* $\langle content \rangle$ 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 $\{ \{ \#1 \} \}$.

Keys for `keyans`, `keyans*` and `keyanspic`

`mark-ans* = { $\langle symbol \rangle$ }` default: `\textasteriskcentered`

Sets the *symbol* to be displayed in the left margin for `\item*` and `\anspic*` for the `keyans`, `keyans*` and `keyanspic` environments when using the key `show-ans`. The “*symbol*” is placed in a box of width equal to the value of `labelwidth` of the environment in which it is executed, 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* = { $\langle left | right | center \rangle$ }` default: `left`

Sets the *aligned* of the “*symbol*” defined by `mark-ans*` key for the `keyans`, `keyans*` and `keyanspic` environments. The “*symbol*” is aligned in a box with the same dimensions of the label box defined by `labelwidth` key of the environment in which it is executed 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.

`wrap-ans* = { $\langle code \{ \#1 \} \text{ more code} \rangle$ }` 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-ans*={\textcolor{red}{\#1}}`.

`wrap-opt = { $\langle code \{ \#1 \} \text{ more code} \rangle$ }` default: `[[\#1]]`

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

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 *mandatory argument* {⟨*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.

*

first answer

2. Text containing our instructions or questions.

(a) Question.

*

second answer
3. Text containing our instructions or questions.

*

third answer

4. Text containing our instructions or questions.

*

fourth answer

6.3 The environment anskey*

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

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

By design the environment cannot be nested but full supports “*verbatim material*” in the `\langle body \rangle` 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 new “*collect code*” c-type argument part of L^AT_EX release 2025-06-01[13]. `\begin{anskey*}` and `\end{anskey*}` must be in different lines and should not appear within verbatim environments or commands. 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, L^AT_EX will return an “error” message if this happens. In a similar way comments “%” or “any character” after `\end{anskey*}` on the same line L^AT_EX 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 `\langle keys \rangle` `write-env`, `overwrite` and `force-eol`. 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}%`.

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}[(key = val)] \item \item[<custom>] \item* \item*[<content>] \end{keyans}`

`keyans*` `\begin{keyans*}[(key = val)] \item \item[<custom>] \item* \item*[<content>] \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[<custom>]` work in the usual and the command `\item(<columns>)` 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 <item content>
  \begin{keyans}[(key = val)]
    \item <item content>
    \item [<custom>] <item content>
    \item* <item content>
    \item* [<content>] <item content>
  \end{keyans}
\end{enumext}
```

```
\begin{enumext}[save-ans=test]
  \item <item content>
  \begin{keyans*}[(key = val)]
    \item <item content>
    \item [<custom>] <item content>
    \item* <item content>
    \item* [<content>] <item content>
  \end{keyans*}
\end{enumext}
```

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

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

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

`\item*` `\item*`
`\item* [<content>]`

The `\item*` and `\item* [<content>]` command “store” the current `<label>` set by `label` key next to the *optional argument* `<content>` in *sequence* and *prop list* `{<store name>}` 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* [<note>] Correct choice
    \miniright
    \includegraphics[scale=0.25]{example-image-a}
    Some text
  \end{keyans}
```

```
\end{keyans}  
\end{enumext}
```

1. Text containing a question.
A) Choice
C) Choice
E) Choice

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


Some text

6.5 The environment keyanspic

```
keyanspic \begin{keyanspic}[\langle 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 `\labels` 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.

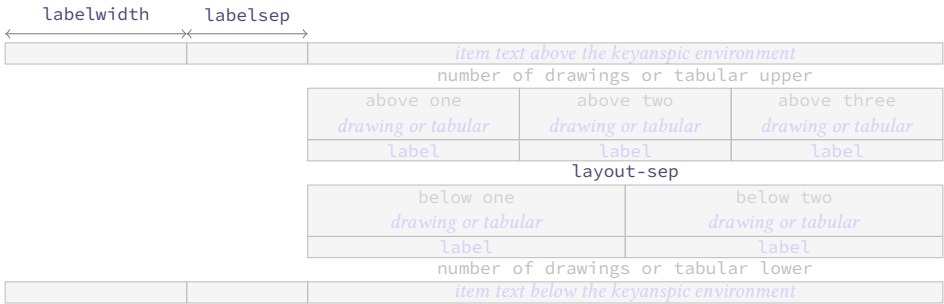


Figure 6: Representation of the `keyanspic` environment with `layout-sty={\langle 3, 2 \rangle}` in `enumext`.

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-ans*`, `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=true,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)




B)



C)




D)




* E) [note]

2. Question with images and labels above.


A)




B)




C)




D)




* E) [note]




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




A)




B)



C)



D)



* 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 <store name> <position>
```

The command `\getkeyans` prints the “*stored content*” in *prop list* `{<store name>}` defined by `save-ans` key in the `<position>` returned by the `show-pos` key.

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

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

6.6.2 The command `\foreachkeyans`

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

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

Options for command

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

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

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

Sets the *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 {<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) ~~ETEX~~e 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:

```
$ pdfdetach -saveall enumext.pdf
```

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


Example 1

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

- | | |
|--|--|
| <p>1. La velocità di $1,00 \times 10^2$ m/s espressa in km/h è:</p> <p><input type="checkbox"/> A 36 km/h.</p> <p><input type="checkbox"/> B 360 km/h.</p> <p><input type="checkbox"/> C 27,8 km/h.</p> <p><input type="checkbox"/> D $3,60 \times 10^8$ km/h.</p> | <p>3. La velocità di $1,00 \times 10^2$ m/s espressa in km/h è:</p> <p><input type="checkbox"/> A 36 km/h.</p> <p><input type="checkbox"/> B 360 km/h.</p> <p><input type="checkbox"/> C 27,8 km/h.</p> <p><input type="checkbox"/> D $3,60 \times 10^8$ km/h.</p> |
| <p>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?</p> <p><input type="checkbox"/> A $1 \text{ \AA} = 1 \times 10^5 \text{ fm}$.</p> <p><input type="checkbox"/> B $1 \text{ \AA} = 1 \times 10^{-5} \text{ fm}$.</p> <p><input type="checkbox"/> C $1 \text{ \AA} = 1 \times 10^{-15} \text{ fm}$.</p> <p><input type="checkbox"/> D $1 \text{ \AA} = 1 \times 10^3 \text{ fm}$.</p> | <p>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?</p> <p><input type="checkbox"/> A $1 \text{ \AA} = 1 \times 10^5 \text{ fm}$.</p> <p><input type="checkbox"/> B $1 \text{ \AA} = 1 \times 10^{-5} \text{ fm}$.</p> <p><input type="checkbox"/> C $1 \text{ \AA} = 1 \times 10^{-15} \text{ fm}$.</p> <p><input type="checkbox"/> D $1 \text{ \AA} = 1 \times 10^3 \text{ fm}$.</p> |

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 \(cross mark\)](#) .

- | | |
|---|---|
| <p>1. La velocità di $1,00 \times 10^2$ m/s espressa in km/h è:</p> <p><input type="checkbox"/> A 36 km/h.</p> <p><input checked="" type="checkbox"/> B 360 km/h.</p> <p><input type="checkbox"/> C 27,8 km/h.</p> <p><input type="checkbox"/> D $3,60 \times 10^8$ km/h.</p> | <p>3. La velocità di $1,00 \times 10^2$ m/s espressa in km/h è:</p> <p><input type="checkbox"/> A 36 km/h.</p> <p><input checked="" type="checkbox"/> B 360 km/h.</p> <p><input type="checkbox"/> C 27,8 km/h.</p> <p><input type="checkbox"/> D $3,60 \times 10^8$ km/h.</p> |
| <p>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?</p> <p><input checked="" type="checkbox"/> A $1 \text{ \AA} = 1 \times 10^5 \text{ fm}$.</p> <p><input type="checkbox"/> B $1 \text{ \AA} = 1 \times 10^{-5} \text{ fm}$.</p> <p><input type="checkbox"/> C $1 \text{ \AA} = 1 \times 10^{-15} \text{ fm}$.</p> <p><input type="checkbox"/> D $1 \text{ \AA} = 1 \times 10^3 \text{ fm}$.</p> | <p>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?</p> <p><input checked="" type="checkbox"/> A $1 \text{ \AA} = 1 \times 10^5 \text{ fm}$.</p> <p><input type="checkbox"/> B $1 \text{ \AA} = 1 \times 10^{-5} \text{ fm}$.</p> <p><input type="checkbox"/> C $1 \text{ \AA} = 1 \times 10^{-15} \text{ fm}$.</p> <p><input type="checkbox"/> D $1 \text{ \AA} = 1 \times 10^3 \text{ fm}$.</p> |

1. B
3. B

- ✖ 2. A
✖ 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. $\angle EDF = 45^\circ$

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

- A

I only

B

II only

C

I and II only
- D

I and III only
- E

I, II, and III

3. Third type of questions

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

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

A

value

B

value

C

value

D

value

E

value

4. Question with image and label below:

A

A

B

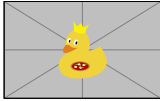
B

A

C

A

D



E

5. Question with image on right side:

- A

value
- B

value
- C

value
- D

correct
- E

value

B

Test keys

1. B, $x = 5$

2. D

3. C, some note
- ✖ 4. E, A duck

✖ 5. D, other note


✖
- ✖


✖

✖


Example 4

A “simple worksheet” using ducks :)

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


Factor $3x + 3y + 3z$

The following questions need to be cuaqtified :)

True False

(a) $\alpha > \delta$

(b) ~~ETX~~ze is cool?

Related to Linux

(a) You use linux?

(b) Usually uses the package manager?

(c) Rate the following package and class

i. xsim-exam

ii. xsim

iii. exsheets

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

1. $(x - 1)^2$

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

3. (a) False

(b) Very True!

4. (a) Yes
- ✖ (b) Yes, dnf

✖ (c) i. doesn't exist for now :(

✖ ii. very good

✖ iii. obsolete

✖
- ✖

✖

✖

✖

✖

Example 5

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

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

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

Example 6

Adapted from the response to [Environment for enumerate environment](#) .

8.5a, KSC 10. sample
A sample
✓ B answer
C sample
D sample

9.5a, KSC 11. sample
A sample
B sample
C sample
✓ D answer

3. sample
A sample
B answer
C sample
D sample







4. sample
A sample
B sample
C sample
D answer







10. B (8.5a, KSC) 12. B (10.5a, KSC)
11. D (9.5a, KSC) 13. D (11.5a, KSC)

8 Tagged PDF examples

This section is just to show the compatibility of [enumext](#) with *tagged* PDF using *lua*latex. 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 *lua*latex-dev the packages [multicol](#), [unicode-math](#), [geometry](#), [graphicx](#), [luamml](#) and [hyperref](#) are required along with the line:

```
\DocumentMetadata
{
  lang = en-US, pdfversion = 2.0, pdfstandard = ua-2, tagging=on,
}
```

- ◆ 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 <i>without</i> a label.
Something	A short one-line description.
Something long	A much longer description. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. If we add <code>align=right</code> it will look like this:

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

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

Description with multi-line labels

The `label` key does not accept *multiline material*, this is where the `wrap-label` 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 <i>without</i> a label.
Something	A short one-line description.
Something long	A much longer description. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.
SoMeThInG LoNg	A much longer description. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.

Final notes

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

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

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

Why list environments?

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

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

Why not random questions and other utilities

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

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] MITTELBAACH, FRANK. “An environment for multicolumn output”. Available from CTAN, <https://www.ctan.org/pkg/multicol>, 2025.
- [4] GONZÁLEZ, PABLO. “scontents - Stores \LaTeX contents in memory or files”. Available from CTAN, <https://www.ctan.org/pkg/scontents>, 2025.
- [5] The \LaTeX Project. “enumerate – Enumerate with redefinable labels”. Available from CTAN, <https://www.ctan.org/pkg/enumerate>, 2025.
- [6] BEZOS, JAVIER. “Customizing lists with the enumitem package”. Available from CTAN, <https://www.ctan.org/pkg/enumitem>, 2025.
- [7] BERRY, KARL. “ $\text{\LaTeX}_{2\epsilon}$: An Unofficial Reference Manual”. Available from CTAN, <https://ctan.org/pkg/latex2e-help-texinfo>, 2025.
- [8] The \LaTeX Project. “Extensive support for hypertext in \LaTeX ”. Available from CTAN, <https://www.ctan.org/pkg/hyperref>, 2025.
- [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>, 2025.
- [11] The \LaTeX Project. “The \LaTeX_3 Interfaces”. Available from CTAN, <https://www.ctan.org/pkg/l3kernel>, 2025.
- [12] The \LaTeX Project. “The $\text{\LaTeX}_{2\epsilon}$ sources”. Available from CTAN, <https://ctan.org/tex-archive/macros/latex/base>, 2025.
- [13] The \LaTeX Project. “ \LaTeX News, Issue 41, June 2025”. Available from CTAN, <https://ctan.org/tex-archive/macros/latex/base>, 2025.
- [14] The \LaTeX Project. “ \LaTeX for authors current version”. Available from CTAN, <https://ctan.org/pkg/latex-base>, 2025.
- [15] GUNDLACH, PATRICK. “The lua-visual-debug package”. Available from CTAN, <https://www.ctan.org/pkg/lua-visual-debug>, 2023.
- [16] LEMVIG, MOGENS. “The shortlst package”. Available from CTAN, <https://www.ctan.org/pkg/shortlst>, 1998.
- [17] NIEDERBERGER, CLEMENS. “tasks – Horizontally columned lists”. Available from CTAN, <https://www.ctan.org/pkg/tasks>, 2022.
- [18] FISCHER, ULRIKE. “tagpdf – \LaTeX kernel code for PDF tagging”. Available from CTAN, <https://www.ctan.org/pkg/tagpdf>, 2025.

- [19] The L^AT_EX Project. “latex-lab – L^AT_EX laboratory”. Available from CTAN, <https://www.ctan.org/pkg/latex-lab>, 2025.
- [20] MITTELBACH, FRANK. “L^AT_EX’s socket management”. Available from CTAN, <https://ctan.org/tex-archive/macros/latex/base>, 2025.

11 Change history

- v1.6 (ctan), 2025-06-21**
 - The `resume`, `resume*` and `series` keys can now be set per level.
 - Fixed bad interaction between `\printkeyans` and the `resume`, `resume*` keys.
- v1.5 (ctan), 2025-06-11**
 - Replacing `\regex_match:` (deprecated) with `\regex_if_match:`.
 - Add keys `beginpenalty`, `midpenalty` and `endpenalty`.
- v1.4 (ctan), 2025-06-09**
 - Improved implementation of the `start` key for *tagged* PDF.
 - Improved implementation of the `ref` key.
 - Fixed the behavior of the `save-sep` key.
 - Fixed the behavior of the `resume*` key.
- v1.3 (ctan), 2025-06-01**
 - Removed dependency on the `scontents` package.
 - The `anskey*` environment has been rewritten using the new `c`-type argument.
- v1.2 (ctan), 2025-03-28**
 - Replace signature (prevent expansion for optional argument).
 - Solve Inconsistent local/global assignment.
- v1.1 (ctan), 2024-11-14**
 - Fixed implementation for `font` and `base-fix` keys.
 - Added new keys for symbol marks.
 - Update and improvements in the internal code.
 - Adjustments in the documentation.
- v1.0 (ctan), 2024-11-01**
 - 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	14
\columnbreak	4, 14	item-join	14
\columnsep	11	item-pos*	14
Commands provide by enumext:		item-star	14
\anskey	11–15	item-sym*	14
\anspic	12–14, 17	Keys for \foreachkeyans provide by enumext:	
\foreachkeyans	19	after	19
\getkeyans	14, 19	before	19
\item*	5–7, 12–14, 16	sep	19
\item	5–7, 10, 11, 14, 16, 17	start	19
\miniright	11	step	19
\printkeyans	6, 12, 19	stop	19
\setenumextmeta	6	wrapper	19
\setenumext	5–7, 12, 13, 16, 20	Keys for anskey* provide by enumext:	
Counters defined by enumext:		break-col	14
enumXiii	4	force-eol	15
enumXii	4	item-join	14
enumXiv	4	item-pos*	14
enumXi	4	item-star	14
enumXviii	4	item-sym*	14
enumXvii	4	overwrite	15
enumXvi	4	write-env	15
enumXv	4	Keys for environments provide by enumext:	
E		above*	9
Environments provide by enumext:		above	9
anskey*	11–15, 24	after	10
enumext*	4–16, 19, 20, 23, 24	align	7, 13, 23–25
enumext	4–17, 19, 20, 23, 24	base-fix	9
keyans*	4–16, 24	before*	10
keyanspic	4, 7, 8, 10–15, 17, 23, 24	before	10
keyans	4–17, 24	beginpenalty	8
Environments:		below*	9
Verbatim	15	below	9
center	5	check-ans	12, 14
description	5, 24	columns-sep	4, 11, 23
enumerate	1, 3, 5, 25	columns	4, 9, 11, 23
figure	5	endpenalty	8
flushleft	5	first	10
flushright	5	font	7, 13
itemize	5, 24	item-pos*	5, 6
list	3, 5, 10, 25	item-sym*	5, 6
minipage	3–5, 8–11, 23, 25	itemindent	9, 10
multicols	3, 4, 11, 23	itemsep	8, 9
quotation	5	label-pos	17
quote	5	label-sep	17
shortenenumerate	5	labelsep	3–7, 9, 11, 23–25
tabbing	5	labelwidth	3, 4, 6, 7, 9, 11, 13, 23–25
table	5	labelwith	5
tasks	5	label	7, 8, 10, 16, 23–25
trivlist	5	labewdith	9
verbatim	5	layout-sep	17
verse	5	layout-sty	17
		layout-top	17

list-indent 3, 9, 10

list-offset 3, 9, 25

listparindent 10

mark-ans* 13, 16, 17

mark-ans 13

mark-pos* 13, 16, 17

mark-pos 13

mark-ref 12

mark-sep* 13, 16, 17

mark-sep 13

midpenalty 8

mini-env 4, 9, 11, 12, 23

mini-right* 7, 11, 12

mini-right 7, 11, 12, 23

mini-sep 4, 11, 12

mode-box 7

no-store 12, 14, 15, 24

noitemsep 9

nosep 9, 24

overwrite 15

parsep 8–10, 17

partopsep 8

ref 4, 8, 23

resume* 7, 10–12

resume 7, 10–12

rightmargin 10

save-ans 4, 6, 10–17, 19, 23, 24

save-key 11, 12, 20

save-ref 4, 7, 12, 14, 15, 19, 23

save-sep 12, 16, 17, 23

series 7, 10–12

show-ans 12, 13, 16, 17, 24

show-length 8

show-pos 12–14, 16, 17, 19

start* 10, 11

start 10, 11

topsep 8, 9, 17

widest 7

wrap-ans* 13, 16, 17

wrap-ans 13

wrap-label* 7, 25

wrap-label 7, 13, 23, 25

wrap-opt 13, 16, 17

write-env 15

L

\label 4

Labels provide by enumext:

 \Alph* 7, 8, 16

 \Roman* 7, 8

 \alph* 7, 8

 \arabic* 7, 8

 \roman* 7, 8

\labelsep 3, 7

\labelwidth 3, 7

\linewidth 11

\listparindent 10

P

Packages:

 enumerate 25

 enumext 1–5, 7, 12, 17, 23–26

 enumitem 3, 4, 25

 fancyvrb 15

 footnotehyper 5

 geometry 23

 graphicx 23

 hyperref 4, 5, 12, 14, 15, 23, 25

 l3keys 7

 l3prop 25

 l3seq 25

 luamml 23

 multicol 1, 2, 4, 23, 25

 scontents 27

 shortlst 5

 tasks 5

 task 6

 unicode-math 23

 xsim 2

\parsep 8

\partopsep 8

R

\raggedcolumns 4

\ref 4

\rightmargin 10

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}[2025-06-01]
```

Now declare the `enumext` package.

```
4 \ProvidesExplPackage {enumext} {2025-06-21} {1.6} {Enumerate exercise sheets}
```

Finally check if the `multicol` package 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}[2025-05-25]
14  }
15 }
```

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.

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

```
\__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

16 \int_new:N \__enumext_level_int
17 \int_new:N \__enumext_level_h_int
18 \int_new:N \__enumext_anskey_level_int
19 \int_new:N \__enumext_keyans_level_int
20 \int_new:N \__enumext_keyans_level_h_int
21 \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_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).

```

22 \bool_new:N \l__enumext_starred_bool
23 \bool_new:N \g__enumext_starred_bool
24 \bool_new:N \l__enumext_starred_first_bool
25 \bool_new:N \l__enumext_standar_bool
26 \bool_new:N \g__enumext_standar_bool
27 \bool_new:N \l__enumext_standar_first_bool
28 \bool_new:N \l__enumext_keyans_env_bool
29 \tl_new:N \g__enumext_start_line_tl
30 \tl_new:N \g__enumext_envir_name_tl
31 \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_counter:Nn` (§13.11) and then modified by the function `__enumext_label_style:Nnn` used by `label` key (§13.14).

```

32 \cs_set_protected:Npn \__enumext_tmp:n #1
33 {
34   \tl_new:c { l__enumext_counter_#1_tl }
35 }
36 \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.)

```

\l__enumext_ref_key_arg_tl
\l__enumext_ref_the_count_tl
\l__enumext_the_counter_X_tl
  \l__enumext_renew_counter_X_tl

```

Internal variables used by `ref` key (§13.14).

```

37 \tl_new:N \l__enumext_ref_key_arg_tl
38 \tl_new:N \l__enumext_ref_the_count_tl
39 \cs_set_protected:Npn \__enumext_tmp:n #1
40 {
41   \tl_new:c { l__enumext_renew_counter_#1_tl }
42   \tl_new:c { l__enumext_the_counter_#1_tl }
43   \tl_set:ce { l__enumext_the_counter_#1_tl } { \exp_not:c { theenumX#1 } }
44 }
45 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }

```

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

```

\l__enumext_series_name_tl
\l__enumext_reset_args_clist
\l__enumext_reset_name_str
\l__enumext_resume_X_bool
\g__enumext_resume_X_int
\g__enumext_not_key_series_X_tl
\g__enumext_series_name_X_tl

```

Internal variables used by `resume`, `resume*` and `series` keys (§13.26).

```

46 \tl_new:N \l__enumext_series_name_tl
47 \clist_new:N \l__enumext_reset_args_clist
48 \str_new:N \l__enumext_reset_name_str
49 \int_new:N \l__enumext_reset_level_int
50 \cs_set_protected:Npn \__enumext_tmp:n #1
51 {
52   \bool_new:c { l__enumext_resume_#1_bool }
53   \int_new:c { g__enumext_resume_#1_int }
54   \tl_new:c { g__enumext_not_key_series_#1_tl }
55   \tl_new:c { l__enumext_series_name_#1_tl }
56 }
57 \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {#1} }

```

(End of definition for `\l__enumext_series_name_tl` 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.

```

58 \dim_new:N \l__enumext_current_widest_dim
59 \tl_new:N \g__enumext_counter_styles_tl
60 \tl_new:N \g__enumext_widest_label_tl
61 \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 (§13.19). The variables `\l__enumext_leftmargin_X_dim` and `\l__enumext_itemindent_X_dim` are used and set by the function `__enumext_calc_hspace:NNNNNNNNNN` (§13.40.1).

```
62 \cs_set_protected:Npn \__enumext_tmp:n #1
63 {
64   \bool_new:c { \l__enumext_leftmargin_tmp_#1_bool }
65   \dim_new:c { \l__enumext_leftmargin_tmp_#1_dim }
66   \dim_new:c { \l__enumext_leftmargin_#1_dim }
67   \dim_new:c { \l__enumext_itemindent_#1_dim }
68 }
69 \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 (§13.23) and `align` key (§13.13).

```
70 \cs_set_protected:Npn \__enumext_tmp:n #1
71 {
72   \skip_new:c { \l__enumext_multicols_above_#1_skip }
73   \skip_new:c { \l__enumext_multicols_below_#1_skip }
74   \skip_new:c { \g__enumext_multicols_right_#1_skip }
75   \str_new:c { \l__enumext_align_label_pos_#1_str }
76 }
77 \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 (§13.24.4) and the keys `mini-right`, `mini-right*`, `mini-env` and `mini-sep` (§13.22, §13.24).

```
78 \int_new:N \g__enumext_minipage_stat_int
79 \skip_new:N \l__enumext_minipage_temp_skip
80 \skip_new:N \l__enumext_minipage_left_skip
81 \skip_new:N \l__enumext_minipage_right_skip
82 \skip_new:N \l__enumext_minipage_after_skip
83 \skip_new:N \g__enumext_minipage_right_skip
84 \skip_new:N \g__enumext_minipage_after_skip
85 \cs_set_protected:Npn \__enumext_tmp:n #1
86 {
87   \dim_new:c { \l__enumext_minipage_left_#1_dim }
88   \bool_new:c { \l__enumext_minipage_active_#1_bool }
89 }
90 \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 (§13.19.1), 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 (§13.21).

```
91 \cs_set_protected:Npn \__enumext_tmp:n #1
92 {
93   \bool_new:c { \l__enumext_wrap_label_#1_bool }
94   \bool_new:c { \l__enumext_wrap_label_opt_#1_bool }
95   \int_new:c { \l__enumext_start_#1_int }
96   \tl_new:c { \l__enumext_fake_item_indent_#1_tl }
97   \tl_new:c { \l__enumext_label_fill_left_#1_tl }
98   \tl_new:c { \l__enumext_label_fill_right_#1_tl }
99   \bool_new:c { \l__enumext_vspace_a_star_#1_bool }
100   \bool_new:c { \l__enumext_vspace_b_star_#1_bool }
101 }
102 \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_current_label_tl
\l__enumext_store_current_opt_arg_tl

```

The variable `\l__enumext_store_active_bool` setting by `save-ans` key (§13.28.1) 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 variables `\l__enumext_store_current_label_tl` and `\l__enumext_store_current_opt_arg_tl` save the *current label* and *optional argument* of `\item*` (§13.39) and `\anspic*` (§13.44.2) for the `keyans`, `keyans*` and `keyanspic` environments.

```

103 \bool_new:N \l__enumext_store_active_bool
104 \tl_new:N \l__enumext_store_name_tl
105 \tl_new:N \g__enumext_store_name_tl
106 \tl_new:N \l__enumext_store_current_label_tl
107 \tl_new:N \l__enumext_store_current_opt_arg_tl

```

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

```

\l__enumext_store_anskey_arg_tl
\l__enumext_store_anskey_env_tl
\l__enumext_write_anskey_env_bool
\l__enumext_write_anskey_env_file_name_tl
\l__enumext_write_anskey_env_file_iow

```

The variable `\l__enumext_store_anskey_arg_tl` save the *argument* of `\anskey` (§13.32) and the variables `\l__enumext_store_anskey_env_tl` save the $\langle body\rangle$ of the environment `anskey*` (§13.33).

The variables `\l__enumext_write_anskey_env_bool`, `\l__enumext_write_anskey_env_file_name_tl` and `\l__enumext_write_anskey_env_file_iow` they are used by the `write-env` and `overwrite` keys in the `anskey*` environment implementation.

```

108 \tl_new:N \l__enumext_store_anskey_arg_tl
109 \tl_new:N \l__enumext_store_anskey_env_tl
110 \bool_new:N \l__enumext_write_anskey_env_bool
111 \tl_new:N \l__enumext_write_anskey_env_file_name_tl
112 \iow_new:N \l__enumext_write_anskey_env_file_iow

```

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

```

\c__enumext_anskey_env_hidden_space_str

```

The `\c__enumext_anskey_env_hidden_space_str` is a constant *string* to used to hide the $\langle forced\ space\rangle$ added by T_EX when recording content in a macro. This *string* contains the *reserved phrase* “ $\%^{\wedge\wedge}\text{Aenumextheol}\%$ ” which is added to the end of the argument stored in *sequence* and *prop list* when the key `force-eol` is false.

```

113 \str_const:Nc \c__enumext_anskey_env_hidden_space_str
114 { \c_percent_str \c_circumflex_str \c_circumflex_str A enumextheol \c_percent_str }

```

(End of definition for `\c__enumext_anskey_env_hidden_space_str`.)

```

\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` (§13.50).

```

115 \tl_new:N \l__enumext_setkey_tmpa_tl
116 \tl_new:N \l__enumext_setkey_tmpb_tl
117 \int_new:N \l__enumext_setkey_tmpa_int
118 \seq_new:N \l__enumext_setkey_tmpa_seq
119 \seq_new:N \l__enumext_setkey_tmpb_seq

```

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

```

\l__enumext_meta_path_tl
\l__enumext_foreach_print_seq
\l__enumext_foreach_name_prop_tl
\l__enumext_foreach_default_keys_tl

```

Internal variables used by the `\printkeyans` command (§13.49) and `\foreachkeyans` command (§13.52).

```

120 \tl_new:N \l__enumext_meta_path_tl
121 \seq_new:N \l__enumext_foreach_print_seq
122 \tl_new:N \l__enumext_foreach_name_prop_tl
123 \tl_new:N \l__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_print_keyans_cmd_bool
\l__enumext_mark_position_str
\l__enumext_mark_position_v_str
\l__enumext_mark_position_viii_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` (§13.49), `show-pos`, `show-ans`, `mark-pos`, `mark-sep` keys (§13.29), `item-sym*` key (§13.37), `save-key` key (§13.29.3) and “*storing structure*”.

```

124 \tl_new:N \l__enumext_print_keyans_starred_tl
125 \bool_new:N \l__enumext_print_keyans_star_bool
126 \bool_new:N \l__enumext_print_keyans_cmd_bool
127 \str_new:N \l__enumext_mark_position_str
128 \str_new:N \l__enumext_mark_position_v_str
129 \str_new:N \l__enumext_mark_position_viii_str
130 \dim_new:N \l__enumext_mark_sep_tmpa_dim
131 \dim_new:N \l__enumext_mark_sep_tmpb_dim
132 \int_new:N \l__enumext_show_pos_tmp_int
133 \tl_new:N \g__enumext_item_symbol_aux_tl
134 \cs_set_protected:Npn \l__enumext_tmp:n #1
135 {
136   \tl_new:c { l__enumext_print_keyans_#1_tl }

```

```

137      \tl_new:c { \__enumext_store_save_key_#1_tl      }
138      \bool_new:c { \__enumext_store_save_key_#1_bool  }
139      \bool_new:c { \__enumext_store_upper_level_#1_bool }
140    }
141    \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {#1} }

```

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

Internal variables used by `keyanspic` environment and `\anspic` command (§13.44.1).

```

142 \seq_new:N \__enumext_anspic_args_seq
143 \dim_new:N \__enumext_anspic_mini_width_dim
144 \int_new:N \__enumext_anspic_above_int
145 \int_new:N \__enumext_anspic_below_int
146 \bool_new:N \__enumext_anspic_label_above_bool
147 \str_new:N \__enumext_anspic_mini_pos_str
148 \box_new:N \__enumext_anspic_label_box
149 \box_new:N \__enumext_anspic_body_box
150 \dim_new:N \__enumext_anspic_label_htdp_dim
151 \dim_new:N \__enumext_anspic_body_htdp_dim

```

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

Internal variables used by “*internal check answer*” mechanism (§13.28.3) used by the `check-ans`, `no-store`, `wrap-ans*` keys and check for starred commands `\item*` in `keyans` and `keyans*` environments and `\anspic*` in `keyanspic` environment.

```

152 \bool_new:N \__enumext_check_answers_bool
153 \bool_new:N \g__enumext_check_ans_key_bool
154 \tl_new:N \__enumext_check_start_line_env_tl
155 \bool_new:N \__enumext_item_wrap_key_bool
156 \int_new:N \g__enumext_check_starred_cmd_int
157 \int_new:N \g__enumext_item_anskey_int
158 \int_new:N \g__enumext_item_number_int
159 \bool_new:N \__enumext_item_number_bool
160 \int_new:N \g__enumext_item_answer_diff_int

```

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

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

```

161 \bool_new:N \__enumext_hyperref_bool
162 \bool_new:N \__enumext_footnotes_key_bool

```

(End of definition for `__enumext_hyperref_bool` and `__enumext_footnotes_key_bool`.)

Internal variables used by `save-ref` key (§13.29). The variables `__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 `__enumext_newlabel_arg_one_tl` and `__enumext_newlabel_arg_two_tl` will be used to form the arguments passed to the function `__enumext_newlabel:nn` (§13.7) and the variable `__enumext_write_aux_file_tl` will be in charge of executing the writing code in the `.aux` file.

```

163 \tl_new:N \__enumext_newlabel_arg_one_tl
164 \tl_new:N \__enumext_newlabel_arg_two_tl
165 \tl_new:N \__enumext_write_aux_file_tl
166 \cs_set_protected:Npn \__enumext_tmp:n #1
167 {
168   \tl_new:c { \__enumext_label_copy_#1_tl }
169 }
170 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }

```

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

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

```

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

```

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


```

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

```

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

```

177 \cs_set_protected:Npn \__enumext_tmp:n #1
178 {
179   \bool_new:c { \l__enumext_item_starred_#1_bool }
180   \int_new:c { \l__enumext_item_column_pos_#1_int }
181   \int_new:c { \g__enumext_item_count_all_#1_int }
182   \int_new:c { \l__enumext_joined_item_#1_int }
183   \int_new:c { \l__enumext_joined_item_aux_#1_int }
184   \int_new:c { \l__enumext_tmpa_#1_int }
185   \dim_new:c { \l__enumext_tmpa_#1_dim }
186   \box_new:c { \l__enumext_item_text_#1_box }
187   \dim_new:c { \l__enumext_joined_width_#1_dim }
188   \dim_new:c { \l__enumext_item_width_#1_dim }
189   \tl_new:c { \g__enumext_item_symbol_aux_#1_tl }
190   \str_new:c { \l__enumext_align_label_#1_str }
191   \bool_new:c { \g__enumext_minipage_active_#1_bool }
192   \box_new:c { \l__enumext_miniright_code_#1_box }
193   \bool_new:c { \g__enumext_minipage_center_#1_bool }
194   \dim_new:c { \g__enumext_minipage_right_#1_dim }
195   \skip_new:c { \g__enumext_minipage_right_#1_skip }
196 }
197 \clist_map_inline:nn { vii, viii } { \__enumext_tmp:n {#1} }

```

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

```
\c__enumext_all_envs_clist
```

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

```

198 \clist_const:Nn \c__enumext_all_envs_clist
199 {
200   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
201   {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
202 }

```

(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 (§13.49) and `\foreachkeyans` command (§13.52).

```

203 \cs_generate_variant:Nn \keys_precompile:nnN { neN }
204 \cs_generate_variant:Nn \seq_use:Nn { NV }

```

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

```
\__enumext_scan_tokens:n
```

The functions `\tl_rescan:n` and `\tl_set_rescan:Nnn` provided by `expl3` doesn't fit the needs of this package because it does not allow catcode changes inside the argument, so verbatim stuff used inside one of `anskey*` environment will not work. Here we create a private copy of `\tex_scantokens:D` which will serve our purposes. See the answer by Ulrich Diez in [How do use {<setup> in \tl_set_rescan:Nnn to replace \scantokens?](#).

```
205 \cs_new_protected:Npn \__enumext_scan_tokens:n #1 { \tex_scantokens:D {#1} }
```

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

```
\__enumext_at_begin_document:n
```

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

```

206 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
207 {
208   \hook_gput_code:nnn {begindocument} {enumext} { #1 }
209 }

```

(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.

```

210 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
211 {
212   \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
213 }
214 \cs_new_protected:Npn \__enumext_before_env:nn #1 #2
215 {
216   \hook_gput_code:nnn {env/#1/before} {enumext} {#2}
217 }

```

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

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

```
218 \cs_new:Nn \__enumext_level:
219 {
220   \int_to_roman:n { \__enumext_level_int }
221 }
```

(End of definition for `__enumext_level:.`)

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

```
222 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
223 {
224   \regex_if_match:nnTF { ^[\+|-]?[\d]+$ } {#1} % $
225   { \prg_return_true: }
226   { \prg_return_false: }
227 }
```

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

`__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.

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

(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.

```
234 \cs_new_protected:Nn \__enumext_unskip_unkern:
235 {
236   \int_case:nnT { \lastnodetype }
237   {
238     { 11 }{ \unskip }
239     { 12 }{ \unkern }
240   }
241 }
```

(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`.
`__enumext_is_on_first_level:`

```
242 \cs_new_protected:Nn \__enumext_is_not_nested:
243 {
244   \str_case:en { \@currenvir }
245   {
246     {enumext}
247     {
248       \tl_set:Nn \l__enumext_envir_name_tl { enumext }
249       \bool_lazy_and:nnT
250       { \bool_not_p:n { \g__enumext_standar_bool } }
251       { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
252       {
253         \bool_gset_true:N \g__enumext_standar_bool
254       }
255     }
256     {enumext*}
257     {
258       \tl_set:Nn \l__enumext_envir_name_tl { enumext* }
259       \bool_lazy_and:nnT
260       { \bool_not_p:n { \g__enumext_starred_bool } }
261       { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
```

```

262         {
263             \bool_gset_true:N \g__enumext_starred_bool
264         }
265     }
266 }
267 }

```

The function `__enumext_is_on_first_level:` will set the variables `\l__enumext_standar_first_bool` (§13.28.1), `\l__enumext_starred_first_bool` (§13.28.1) 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.

```

268 \cs_new_protected:Nn \__enumext_is_on_first_level:
269 {
270     \bool_lazy_all:nT
271     {
272         { \bool_if_p:N \g__enumext_standar_bool }
273         { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
274         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
275     }
276     {
277         \bool_set_true:N \l__enumext_standar_first_bool
278         \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
279         \tl_gset:Nn \g__enumext_start_line_tl
280             {
281                 on~line~\exp_not:V \inputlineno
282             }
283     }
284     \bool_lazy_all:nT
285     {
286         { \bool_if_p:N \g__enumext_starred_bool }
287         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
288         { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
289     }
290     {
291         \bool_set_true:N \l__enumext_starred_first_bool
292         \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
293         \tl_gset:Nn \g__enumext_start_line_tl
294             {
295                 on~line~\exp_not:V \inputlineno
296             }
297     }
298 }

```

(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.

```

299 \cs_new_protected:Nn \__enumext_keyans_name_and_start:
300 {
301     \str_case:en { \@currenvir }
302     {
303         {keyans}
304         {
305             \tl_set:Nn \l__enumext_envir_name_tl { keyans }
306             \tl_set:Nn \l__enumext_check_start_line_env_tl
307                 {
308                     in~'keyans'~start~on~line~\exp_not:V \inputlineno
309                 }
310         }
311         {keyans*}
312         {
313             \tl_set:Nn \l__enumext_envir_name_tl { keyans* }
314             \tl_set:Nn \l__enumext_check_start_line_env_tl
315                 {
316                     in~'keyans*'~start~on~line~\exp_not:V \inputlineno
317                 }
318         }
319         {keyanspic}
320         {

```

```

321         \tl_set:Nn \l__enumext_envir_name_tl { keyanspic }
322         \tl_set:Nc \l__enumext_check_start_line_env_tl
323             {
324             in~'keyanspic'~start~on~line~\exp_not:V \inputlineno
325             }
326     }
327 }
328 }

```

(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.

```

329 \cs_new_protected:Nn \__enumext_reset_global_vars:
330 {
331     \__enumext_reset_global_int:
332     \__enumext_reset_global_bool:
333     \__enumext_reset_global_tl:
334 }
335 \cs_new_protected:Nn \__enumext_reset_global_int:
336 {
337     \int_gzero:N \g__enumext_item_number_int
338     \int_gzero:N \g__enumext_item_anskey_int
339     \int_gzero:N \g__enumext_item_answer_diff_int
340 }
341 \cs_new_protected:Nn \__enumext_reset_global_bool:
342 {
343     \bool_gset_false:N \g__enumext_check_ans_key_bool
344     \bool_gset_false:N \g__enumext_standar_bool
345     \bool_gset_false:N \g__enumext_starred_bool
346 }
347 \cs_new_protected:Nn \__enumext_reset_global_tl:
348 {
349     \tl_gclear:N \g__enumext_store_name_tl
350     \tl_gclear:N \g__enumext_start_line_tl
351     \tl_gclear:N \g__enumext_envir_name_tl
352 }

```

(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.

```

353 \cs_new_protected:Nn \__enumext_log_global_vars:
354 {
355     \msg_log:nneeee { enumext } { prop-seq-int-hook }
356     { \g__enumext_store_name_tl }
357     { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
358     { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
359     { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
360 }

```

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.

```

361 \cs_new_protected:Nn \__enumext_log_answer_vars:
362 {
363     \msg_log:nneeee { enumext } { item-answer-hook }
364     { \int_use:N \g__enumext_item_number_int }
365     { \int_use:N \g__enumext_item_anskey_int }
366     { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int } }
367 }

```

(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 \LaTeX has the following plain form:

```
\list{<arg one>}{<arg two>}
  \item[<opt>]
\endlist
```

And `minipage` environment provided by \LaTeX 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`[19]).

```
\__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.

```
368 \__enumext_at_begin_document:n
369 {
370   \cs_new_eq:NN \__enumext_start_list:nn \list
371   \cs_new_eq:NN \__enumext_stop_list: \endlist
372   \NewCommandCopy \__enumext_item_std:w \item
373 }
```

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

```
374 \__enumext_at_begin_document:n
375 {
376   \cs_new_eq:NN \__enumext_minipage:w \minipage
377   \cs_new_eq:NN \__enumext_endminipage: \endminipage
378 }
```

(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.

```
379 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
380 \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 `\l__enumext_footnotes_key_bool` to “true”.

```
381 \cs_new_protected:Nn \__enumext_after_hyperref:
382 {
383   \IfPackageLoadedT { hyperref }
384   {
385     \msg_info:nnn { enumext } { package-load } { hyperref }
386     \bool_set_true:N \l__enumext_hyperref_bool
387     \IfHyperBoolean{hyperfootnotes}
388     {
389       \bool_set_true:N \l__enumext_footnotes_key_bool
390     }
391     { }
392   }
```

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`.

```
393   \bool_if:NT \l__enumext_footnotes_key_bool
394   {
395     \IfPackageLoadedTF { footnotehyper }
396     {
397       \msg_info:nnn { enumext } { package-load } { footnotehyper }
398     }
399     {
400       \bool_set_false:N \l__enumext_footnotes_key_bool
```

```

401     }
402 }

```

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.

```

403 \bool_if:NTF \l__enumext_hyperref_bool
404 {
405   \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
406   \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
407 }
408 {
409   \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
410   \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
411 }
412 }

```

(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.

```

413 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
414 {
415   \protected@write \@auxout { }
416   {
417     \token_to_str:N \newlabel {#1}
418     {
419       {#2}
420       \bool_if:NT \l__enumext_hyperref_bool
421       { { \thepage } {#2} {#1} }
422       { }
423     }
424   }
425   \__enumext_hypertarget:nn {#1} { }
426   \__enumext_phantomsection:
427 }

```

(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.

```

428 \cs_new_protected:Nn \__enumext_footnotetext:nn
429 {
430   \footnotetext[#1]{#2}
431 }
432 \cs_new_protected:Nn \__enumext_renew_footnote:
433 {
434   \RenewDocumentCommand \footnote { o +m }
435   {
436     \tl_if_novalue:nTF {##1}
437     {
438       \stepcounter{footnote}
439       \int_gset_eq:Nc \g__enumext_footnote_standar_int { c@footnote }
440     }
441     {
442       \int_gset:Nn \g__enumext_footnote_standar_int { ##1 }
443     }
444     \footnotemark [ \g__enumext_footnote_standar_int ]
445     \seq_gput_right:Nn \g__enumext_footnote_standar_arg_seq { ##2 }
446     \seq_gput_right:NV
447     \g__enumext_footnote_standar_int_seq \g__enumext_footnote_standar_int

```



```

448     }
449   }
450   \cs_new_protected:Nn \__enumext_print_footnote:
451   {
452     \seq_if_empty:NF \g__enumext_footnote_standar_int_seq
453     {
454       \seq_map_pairwise_function:NNN
455       \g__enumext_footnote_standar_int_seq
456       \g__enumext_footnote_standar_arg_seq
457       \__enumext_footnotetext:nn
458     }
459     \seq_gclear:N \g__enumext_footnote_standar_arg_seq
460     \seq_gclear:N \g__enumext_footnote_standar_int_seq
461   }

```

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.

```

462   \cs_new_protected:Nn \__enumext_renew_footnote_mini:
463   {
464     \RenewDocumentCommand \footnote { o +m }
465     {
466       \tl_if_novalue:nTF {##1}
467       {
468         \stepcounter{footnote}
469         \int_gset_eq:Nc \g__enumext_footnote_starred_int { c@footnote }
470       }
471       {
472         \int_gset:Nn \g__enumext_footnote_starred_int { ##1 }
473       }
474       \footnotemark [ \g__enumext_footnote_starred_int ]
475       \seq_gput_right:Nn \g__enumext_footnote_starred_arg_seq { ##2 }
476       \seq_gput_right:NV
477       \g__enumext_footnote_starred_int_seq \g__enumext_footnote_starred_int
478     }
479   }
480   \cs_new_protected:Nn \__enumext_print_footnote_mini:
481   {
482     \seq_if_empty:NF \g__enumext_footnote_starred_int_seq
483     {
484       \seq_map_pairwise_function:NNN
485       \g__enumext_footnote_starred_int_seq
486       \g__enumext_footnote_starred_arg_seq
487       \__enumext_footnotetext:nn
488     }
489     \seq_gclear:N \g__enumext_footnote_starred_arg_seq
490     \seq_gclear:N \g__enumext_footnote_starred_int_seq
491   }

```

(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.

```

492   \cs_new_protected:Nn \__enumext_renew_footnote_standar:
493   {
494     \bool_if:NT \g__enumext_standar_bool
495     {
496       \IfDocumentMetadataTF
497       {
498         \__enumext_renew_footnote:
499       }
500       {
501         \bool_if:NF \l__enumext_footnotes_key_bool
502         {
503           \__enumext_renew_footnote:
504         }
505       }
506     }
507   }
508   \cs_new_protected:Nn \__enumext_print_footnote_standar:
509   {

```

```

510 \bool_if:NT \g__enumext_standar_bool
511 {
512   \IfDocumentMetadataTF
513   {
514     \__enumext_print_footnote:
515   }
516   {
517     \bool_if:NF \l__enumext_footnotes_key_bool
518     {
519       \__enumext_print_footnote:
520     }
521   }
522 }
523 }

```

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.

```

524 \cs_new_protected:Nn \__enumext_renew_footnote_starred:
525 {
526   \IfDocumentMetadataTF
527   {
528     \__enumext_renew_footnote_mini:
529   }
530   {
531     \bool_if:NF \l__enumext_footnotes_key_bool
532     {
533       \__enumext_renew_footnote_mini:
534     }
535   }
536 }
537 \cs_new_protected:Nn \__enumext_print_footnote_starred:
538 {
539   \IfDocumentMetadataTF
540   {
541     \__enumext_print_footnote_mini:
542   }
543   {
544     \bool_if:NF \l__enumext_footnotes_key_bool
545     {
546       \__enumext_print_footnote_mini:
547     }
548   }
549 }

```

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

```

550 \__enumext_after_env:nn { enumext* }
551 {
552   \__enumext_print_footnote_starred:
553 }
554 \__enumext_after_env:nn { keyans* }
555 {
556   \__enumext_print_footnote_starred:
557 }

```

(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 (§13.41) and `__enumext_safe_exec_vii:` in the `enumext*` environment definition (§13.46)

```

558 \cs_new_protected:Nn \__enumext_internal_mini_page:
559 {
560   \int_compare:nNnT { \l__enumext_level_int } = { 0 }
561   {

```

```

562 \DeclareDocumentEnvironment{__enumext_mini_page}{ m }
563 {
564     \__enumext_renew_footnote_standar:
565     \__enumext_minipage:w [ t ] { ##1 }
566     \legacy_if_gset_false:n { @minipage }
567     \skip_vertical:N \c_zero_skip
568 }
569 {
570     \skip_vertical:N \c_zero_skip
571     \__enumext_endminipage:
572     \__enumext_print_footnote_standar:
573 }
574 }
575 }

```

(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.

```

576 \dim_zero_new:N \itemwidth

```

13.11 Definition of counters

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.

```

enumXi
enumXii
enumXiii
enumXiv
enumXv
enumXvi
enumXvii
enumXviii
577 \cs_new_protected:Npn \__enumext_define_counter:Nn #1 #2
578 {
579     \cs_if_exist:cTF { c@ #2 }
580     { \msg_fatal:nnn { enumext } { counters }{ #2 } }
581     {
582         \tl_set:Nn #1 { #2 }
583         \newcounter { #2 }
584     }
585 }

```

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.

```

586 \__enumext_define_counter:Nn \__enumext_counter_i_tl { enumXi }
587 \__enumext_define_counter:Nn \__enumext_counter_ii_tl { enumXii }
588 \__enumext_define_counter:Nn \__enumext_counter_iii_tl { enumXiii }
589 \__enumext_define_counter:Nn \__enumext_counter_iv_tl { enumXiv }
590 \__enumext_define_counter:Nn \__enumext_counter_v_tl { enumXv }
591 \__enumext_define_counter:Nn \__enumext_counter_vi_tl { enumXvi }
592 \__enumext_define_counter:Nn \__enumext_counter_vii_tl { enumXvii }
593 \__enumext_define_counter:Nn \__enumext_counter_viii_tl { enumXviii }

```

(End of definition for `__enumext_define_counter: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.

- 🔗 Direct support for this is provided since L^AT_EX release 2025-06-01[13], but we will keep the original implementation so as not to hinder the internal “*label and ref*” system.

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.

```

594 \cs_new_protected:Npn \__enumext_register_default_label_wd:Nn #1 #2
595 {
596     \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
597     \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
598 }
599 \__enumext_register_default_label_wd:Nn \arabic { 0 }
600 \__enumext_register_default_label_wd:Nn \Alph { M }

```

```

601 \__enumext_register_default_label_wd:Nn \alph { m }
602 \__enumext_register_default_label_wd:Nn \Roman { VIII }
603 \__enumext_register_default_label_wd:Nn \roman { viii }

```

(End of definition for __enumext_register_default_label_wd: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.

```

604 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
605 {
606   \hbox_set:Nn \l__enumext_label_width_by_box {#2}
607   \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
608 }
609 \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.

```

610 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
611 {
612   \tl_clear_new:N #1
613   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
614   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
615   \tl_map_inline:Nn \g__enumext_counter_styles_tl
616   {
617     \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
618     \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
619     { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
620   }
621   \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
622   { \tl_use:N \g__enumext_widest_label_tl }
623   \tl_set_eq:cN { the #2 } #1
624 }
625 \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 (§13.36). 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.

```

626 \cs_set_protected:Npn \__enumext_tmp:n #1
627 {
628   \keys_define:nn { enumext / #1 }
629   {
630     mode-box .bool_set:N = \l__enumext_mode_box_bool,
631     mode-box .initial:n = false,
632     mode-box .value_forbidden:n = true,
633   }
634 }
635 \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.

```

636 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
637 {
638   \keys_define:nn { enumext / #1 }
639   {
640     font .tl_set:c = { l__enumext_label_font_style_#2_tl },
641     font .value_required:n = true,
642     labelsep .dim_set:c = { l__enumext_labelsep_#2_dim },
643     labelsep .initial:n = {0.3333em},

```

```

644     labelsep .value_required:n = true,
645     labelwidth .dim_set:c = { l__enumext_labelwidth_#2_dim },
646     labelwidth .value_required:n = true,
647     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
648     wrap-label .initial:n = {##1},
649     wrap-label .value_required:n = true,
650     wrap-label* .code:n = {
651         \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
652         \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
653     },
654     wrap-label* .value_required:n = true,
655 }
656 }
657 \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`.

```

658 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
659 {
660     \keys_define:nn { enumext / #1 }
661     {
662         align .choice:,
663         align / left .code:n =
664             {
665                 \tl_clear:c { l__enumext_label_fill_left_#2_tl }
666                 \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
667                 \str_set:cn { l__enumext_align_label_pos_#2_str } { l }
668             },
669         align / right .code:n =
670             {
671                 \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
672                 \tl_clear:c { l__enumext_label_fill_right_#2_tl }
673                 \str_set:cn { l__enumext_align_label_pos_#2_str } { r }
674             },
675         align / center .code:n =
676             {
677                 \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
678                 \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
679                 \str_set:cn { l__enumext_align_label_pos_#2_str } { c }
680             },
681         align / unknown .code:n =
682             \msg_error:nneee { enumext } { unknown-choice }
683             { align } { left,~right,~ center } { \exp_not:n {##1} },
684         align .initial:n = left,
685         align .value_required:n = true,
686     }
687 }
688 \clist_map_inline:nn
689 {
690     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
691 }
692 { \__enumext_tmp:nn #1 }
693 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
694 {
695     \keys_define:nn { enumext / #1 }
696     {
697         align .choice:,
698         align / left .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
699         align / right .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
700         align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
701         align / unknown .code:n =
702             \msg_error:nneee { enumext } { unknown-choice }
703             { align } { left,~right,~ center } { \exp_not:n {##1} },
704         align .initial:n = left,
705         align .value_required:n = true,
706     }
707 }
708 \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 $\langle label \rangle$, the value of the variables $\backslash_enumext_label_X_tl$, the default values for $\backslash labelwidth$ and the “*label and ref*” system.

13.14.1 Define and set label and ref keys for enumext environment

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

```

label
ref
\l__enumext_label_i_tl
\l__enumext_label_ii_tl
\l__enumext_label_iii_tl
\l__enumext_label_iv_tl
709 \cs_set_protected:Npn \l__enumext_tmp:nnn #1 #2 #3
710 {
711   \keys_define:nn { enumext / #1 }
712   {
713     label .code:n = {
714       \l__enumext_label_style:cvn { \l__enumext_label_#2_tl }
715       { \l__enumext_counter_#2_tl } {##1}
716       \dim_set_eq:cN { \l__enumext_labelwidth_#2_dim }
717       \l__enumext_current_widest_dim
718     },
719     label .initial:n = #3,
720     label .value_required:n = true,
721     ref .code:n = \l__enumext_standar_ref:n {##1},
722     ref .value_required:n = true,
723   }
724 }
725 \l__enumext_tmp:nnn { level-1 } { i } { \arabic*. }
726 \l__enumext_tmp:nnn { level-2 } { ii } { (\alph*. ) }
727 \l__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
728 \l__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }

```

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

```

\l__enumext_standar_ref:n
\l__enumext_standar_ref:

```

The `\l__enumext_standar_ref:n` function will first pass the key *argument* `ref` to the variable $\backslash_enumext_ref_key_arg_tl$ and analyze its state, if it is not *empty* it will set a copy of of the *current counter style* save in $\backslash_enumext_the_counter_X_tl$ to $\backslash_enumext_ref_the_count_tl$ and then set the variable $\backslash_enumext_renew_counter_X_tl$ which will modify $\backslash theenumX$.

```

729 \cs_new_protected:Npn \l__enumext_standar_ref:n #1
730 {
731   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
732   \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
733   {
734     \msg_error:nnn { enumext } { key-ref-empty } { enumext }
735   }
736   {
737     \tl_set_eq:Nc \l__enumext_ref_the_count_tl
738     {
739       \l__enumext_the_counter_ \l__enumext_level: _tl
740     }
741     \tl_set:ce { \l__enumext_renew_counter_ \l__enumext_level: _tl }
742     {
743       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
744       { \exp_not:V \l__enumext_ref_key_arg_tl }
745     }
746   }
747 }

```

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

```

748 \cs_new_protected:Npn \l__enumext_standar_ref:
749 {
750   \tl_if_empty:cF { \l__enumext_renew_counter_ \l__enumext_level: _tl }
751   {
752     \tl_use:c { \l__enumext_renew_counter_ \l__enumext_level: _tl }
753   }
754 }

```

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

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

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

```

\l__enumext_label_vii_tl 755 \cs_set_protected:Npn \l__enumext_tmp:nnn #1 #2 #3
\l__enumext_label_viii_tl 756 {
757   \keys_define:nn { enumext / #1 }
758   {
759     label .code:n = {
760       \__enumext_label_style:cvn { \l__enumext_label_#2_tl }
761       { \l__enumext_counter_#2_tl } {##1}
762       \dim_set_eq:cN { \l__enumext_labelwidth_#2_dim }
763       \l__enumext_current_widest_dim
764     },
765     label .initial:n = #3,
766     label .value_required:n = true,
767     ref .code:n = \l__enumext_starred_ref:n {##1},
768     ref .value_required:n = true,
769   }
770 }
771 \l__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
772 \l__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:

```

773 \cs_new_protected:Npn \l__enumext_starred_ref:n #1
774 {
775   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
776   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
777   {
778     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
779     {
780       \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
781     }
782     {
783       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
784       \tl_set:Ne \l__enumext_renew_counter_vii_tl
785       {
786         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V
787       }
788     }
789   }
790   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
791   {
792     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
793     {
794       \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
795     }
796     {
797       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
798       \tl_set:Ne \l__enumext_renew_counter_viii_tl
799       {
800         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V
801       }
802     }
803   }
804 }

```

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.

```

805 \cs_new_protected:Nn \l__enumext_starred_ref:
806 {
807   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
808   {
809     \tl_if_empty:NF \l__enumext_renew_counter_vii_tl
810     {
811       \tl_use:N \l__enumext_renew_counter_vii_tl
812     }
813   }
814   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
815   {

```

```

816         \tl_if_empty:NF \l__enumext_renew_counter_viii_tl
817         {
818             \tl_use:N \l__enumext_renew_counter_viii_tl
819         }
820     }
821 }

```

(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` if it has not been established and `ref` key. The `keyanspic` environment use the same `<label>` as the `keyans` environment.

```

\__enumext_label_v_tl
\__enumext_label_vi_tl
822 \keys_define:nn { enumext / keyans }
823 {
824     label .code:n = {
825         \__enumext_label_style:cnv { \__enumext_label_v_tl }
826         { \__enumext_counter_v_tl } {#1}
827         \__enumext_label_style:cnv { \__enumext_label_vi_tl }
828         { \__enumext_counter_vi_tl } {#1}
829         \dim_set_eq:NN
830         \l__enumext_labelwidth_v_dim \l__enumext_current_widest_dim
831     },
832     label .initial:n = \Alph*,
833     label .value_required:n = true,
834     ref .code:n = \__enumext_keyans_ref:n {#1},
835     ref .value_required:n = true,
836 }

```

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

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

```

\__enumext_keyans_ref:n
\__enumext_keyans_ref:
837 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
838 {
839     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
840     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
841     {
842         \msg_error:nnn { enumext } { key-ref-empty } { keyans }
843     }
844     {
845         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
846         \tl_put_right:Ne \l__enumext_renew_counter_v_tl
847         {
848             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V \l__
849         }
850     }
851 }

```

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

```

852 \cs_new_protected:Nn \__enumext_keyans_ref:
853 {
854     \tl_if_empty:NF \l__enumext_renew_counter_v_tl
855     {
856         \tl_use:N \l__enumext_renew_counter_v_tl
857     }
858 }

```

(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.

```

859 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
860 {

```

```

861     \__enumext_if_is_int:nTF { #3 }
862     {
863         \int_set:Nn #2 {#3}
864     }
865     {
866         \regex_if_match:nVT { \c{Alph} | \c{alph} } {#1}
867         { \int_set:Nn #2 { \int_from_alph:n {#3} } }
868         \regex_if_match:nVT { \c{Roman} | \c{roman} } {#1}
869         { \int_set:Nn #2 { \int_from_roman:n {#3} } }
870     }
871 }
872 \cs_generate_variant:Nn \__enumext_start_from:NNn { ccn, cce }

```

(End of definition for __enumext_start_from:NNn.)

```

\__enumext_widest_from:nNNn
\__enumext_widest_from:nccn

```

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

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

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

```

873 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
874 {
875     \__enumext_if_is_int:nTF {#4}
876     {
877         \setcounter{enumX#1} { #4 }
878     }
879     {
880         \regex_if_match:nVT { \c{Alph} | \c{alph} } {#2}
881         { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
882         \regex_if_match:nVT { \c{Roman} | \c{roman} } {#2}
883         { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
884     }
885     \__enumext_label_width_by_box:cv
886     { \__enumext_labelwidth_#1_dim } { \__enumext_label_#1_tl }
887 }
888 \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
889 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
890 {
891     \keys_define:nn { enumext / #1 }
892     {
893         start* .code:n = {
894             \__enumext_start_from:ccn
895             { \__enumext_label_#2_tl }
896             { \__enumext_start_#2_int } {##1}
897         },
898         start* .value_required:n = true,
899         start .code:n = {
900             \__enumext_start_from:cce
901             { \__enumext_label_#2_tl }
902             { \__enumext_start_#2_int } { \int_eval:n {##1} }
903         },
904         start .initial:n = 1,
905         start .value_required:n = true,
906         widest .code:n = {
907             \__enumext_widest_from:nccn {#2}
908             { \__enumext_label_#2_tl }
909             { \__enumext_labelwidth_#2_dim } {##1}
910         },
911         widest .value_required:n = true,
912     }
913 }
914 \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 penaltys

`beginpenalty`
`midpenalty`
`endpenalty`

The three parameters `\@beginparpenalty`, `\@itempenalty` and `\@endparpenalty` work together to ensure that list environments look good, avoiding unsightly page breaks that can break the flow of the `list`, so it's a good idea to have a `(keys)` to access these.

```

915 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
916 {
917   \keys_define:nn { enumext / #1 }
918   {
919     beginpenalty .int_set:c = { l__enumext_beginparpenalty_#2_int },
920     beginpenalty .initial:n = -51,
921     beginpenalty .value_required:n = true,
922     midpenalty .int_set:c = { l__enumext_itempenalty_#2_int },
923     midpenalty .initial:n = -51,
924     midpenalty .value_required:n = true,
925     endpenalty .int_set:c = { l__enumext_endparpenalty_#2_int },
926     endpenalty .initial:n = -51,
927     endpenalty .value_required:n = true,
928   }
929 }
930 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

(End of definition for `beginpenalty`, `midpenalty`, and `endpenalty`.)

13.17 Setting keys for vertical spaces

`topsep`
`partopsep`
`parsep`
`noitemsep`
`nosep`

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

```

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

```

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

```

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

```

```

971 { 4.0pt plus 2.0pt minus 1.0pt }
972 \__enumext_tmp:nnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
973 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
974 { 2.0pt plus 1.0pt minus 1.0pt }

```

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

13.18 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.

We define the key `base-fix` only for the “first level” of `enumext` environment.

```

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

```

The function `__enumext_nested_base_line_fix:` passed to the `__enumext_parse_keys:n` function in the definition of the `enumext` environment (§13.41) will be responsible for applying the *baseline correction* and adjusting the *keys* for the `enumext` environment and the `\printkeyans` with *starred argument* ‘*’ (§13.49).

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 `\l__enumext_print_keyans_star_bool` set by the `\printkeyans` command is false and the variable `\l__enumext_base_line_fix_bool` is true.

We set the values of the keys `topsep`, `above` and `above*` for the “first level” of `enumext` environment equal to `0pt` and finally set the variable `\l__enumext_base_line_fix_bool` to false.

```

981 \cs_new_protected:Nn \__enumext_nested_base_line_fix:
982 {
983   \bool_lazy_all:nT
984   {
985     { \bool_if_p:N \l__enumext_starred_first_bool }
986     { \bool_if_p:N \l__enumext_base_line_fix_bool }
987     { \bool_not_p:n { \l__enumext_print_keyans_star_bool } }
988   }
989   {
990     \mode_leave_vertical:
991     \vspace { -\dim_eval:n { \baselineskip + \parsep } }
992     \keys_set:nn { enumext / level-1 }
993     {
994       topsep = 0pt, above = 0pt, above* = 0pt,
995     }
996   }
997 }

```

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

```

997 \bool_lazy_and:nnT
998 { \bool_if_p:N \l__enumext_starred_first_bool }
999 { \bool_if_p:N \l__enumext_print_keyans_star_bool }
1000 {
1001   \mode_leave_vertical:
1002   \skip_vertical:n { -\baselineskip }
1003   \skip_vertical:N \c_zero_skip
1004   \keys_set:nn { enumext / level-1 }
1005   {
1006     topsep = 0pt, above = 0pt, above* = 0pt,
1007   }
1008 }
1009 \bool_set_false:N \l__enumext_base_line_fix_bool
1010 }

```

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

13.19 Setting keys for horizontal spaces

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

```

1011 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1012 {
1013   \keys_define:nn { enumext / #1 }
1014   {
1015     itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
1016     itemindent .value_required:n = true,
1017     rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
1018     rightmargin .value_required:n = true,
1019     listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
1020     listparindent .value_required:n = true,
1021     list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
1022     list-offset .value_required:n = true,
1023     list-indent .code:n =
1024       \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
1025       \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
1026     list-indent .value_required:n = true,
1027   }
1028 }
1029 \clist_map_inline:nn
1030 {
1031   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
1032 }
1033 { \__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.

```

1034 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1035 {
1036   \keys_define:nn { enumext / #1 }
1037   {
1038     itemindent .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
1039     itemindent .value_required:n = true,
1040     rightmargin .dim_set:c = { l__enumext_rightmargin_#2_dim },
1041     rightmargin .value_required:n = true,
1042     listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
1043     listparindent .value_required:n = true,
1044     list-offset .dim_set:c = { l__enumext_listoffset_#2_dim },
1045     list-offset .value_required:n = true,
1046     list-indent .meta:n = { list-offset = ##1 },
1047     list-indent .value_required:n = true,
1048   }
1049 }
1050 \clist_map_inline:nn
1051 {
1052   {enumext*}{vii}, {keyans*}{viii}
1053 }
1054 { \__enumext_tmp:nn #1 }

```

13.19.1 Functions for setting the fake `itemindent`

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

```

1055 \cs_set_protected:Nn \__enumext_fake_item_indent:
1056 {
1057   \dim_compare:nNnT
1058     { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
1059     >
1060     { \c_zero_dim }
1061   {
1062     \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
1063     {
1064       \exp_not:N \mode_leave_vertical:
1065       \exp_not:n { \skip_horizontal:n }
1066       { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }

```



```

1067         \exp_not:N \ignorespaces
1068     }
1069 }
1070 }
1071 \cs_set_protected:Nn \__enumext_keyans_fake_item_indent:
1072 {
1073     \dim_compare:nNnT
1074     { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
1075     {
1076         \tl_set:Nc \l__enumext_fake_item_indent_v_tl
1077         {
1078             \exp_not:N \mode_leave_vertical:
1079             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
1080             \exp_not:N \ignorespaces
1081         }
1082     }
1083 }
1084 \cs_set_protected:Nn \__enumext_fake_item_indent_vii:
1085 {
1086     \dim_compare:nNnT
1087     { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
1088     {
1089         \tl_set:Nc \l__enumext_fake_item_indent_vii_tl
1090         {
1091             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
1092             \exp_not:N \ignorespaces
1093         }
1094     }
1095 }
1096 \cs_set_protected:Nn \__enumext_fake_item_indent_viii:
1097 {
1098     \dim_compare:nNnT
1099     { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
1100     {
1101         \tl_set:Nc \l__enumext_fake_item_indent_viii_tl
1102         {
1103             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
1104             \exp_not:N \ignorespaces
1105         }
1106     }
1107 }

```

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

13.20 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.

```

1108 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1109 {
1110     \keys_define:nn { enumext / #1 }
1111     {
1112         show-length .bool_set:c = { \l__enumext_show_length_#2_bool },
1113         show-length .initial:n = false,
1114     }
1115 }
1116 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

13.21 Setting before, after and first keys

before

before*

after

first

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

```

1117 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1118 {
1119     \keys_define:nn { enumext / #1 }
1120     {
1121         before .tl_set:c = { \l__enumext_before_no_starred_key_#2_tl },
1122         before .value_required:n = true,

```

```

1123     before* .tl_set:c = { l__enumext_before_starred_key_#2_tl },
1124     before* .value_required:n = true,
1125     after .tl_set:c = { l__enumext_after_stop_list_#2_tl },
1126     after .value_required:n = true,
1127     first .tl_set:c = { l__enumext_after_list_args_#2_tl },
1128     first .value_required:n = true,
1129   }
1130 }
1131 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

13.21.1 Functions for before, after and first keys in enumext

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

```

1132 \cs_new_protected:Nn \__enumext_before_args_exec:
1133 {
1134   \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
1135 }

```

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⟩}`

```

1136 \cs_new_protected:Nn \__enumext_before_keys_exec:
1137 {
1138   \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
1139 }

```

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

```

1140 \cs_new_protected:Nn \__enumext_after_stop_list:
1141 {
1142   \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
1143 }

```

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.`

```

1144 \cs_new_protected:Nn \__enumext_after_args_exec:
1145 {
1146   \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
1147 }

```

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

13.21.2 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:
1148 \cs_new_protected:Nn \__enumext_before_args_exec_v:
1149 {
1150   \tl_use:N \l__enumext_before_starred_key_v_tl
1151 }
1152 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
1153 {
1154   \tl_use:N \l__enumext_before_no_starred_key_v_tl
1155 }
1156 \cs_new_protected:Nn \__enumext_after_stop_list_v:
1157 {
1158   \tl_use:N \l__enumext_after_stop_list_v_tl
1159 }
1160 \cs_new_protected:Nn \__enumext_after_args_exec_v:
1161 {
1162   \tl_use:N \l__enumext_after_list_args_v_tl
1163 }

```

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

13.21.3 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:
1164 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
1165 {
1166   \tl_use:N \l__enumext_before_starred_key_vii_tl
1167 }
1168 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
1169 {
1170   \tl_use:N \l__enumext_before_starred_key_viii_tl
1171 }
1172 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
1173 {
1174   \tl_use:N \l__enumext_before_no_starred_key_vii_tl
1175 }
1176 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
1177 {
1178   \tl_use:N \l__enumext_before_no_starred_key_viii_tl
1179 }
1180 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
1181 {
1182   \tl_use:N \l__enumext_after_stop_list_vii_tl
1183 }
1184 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
1185 {
1186   \tl_use:N \l__enumext_after_stop_list_viii_tl
1187 }
1188 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
1189 {
1190   \tl_use:N \l__enumext_after_list_args_vii_tl
1191 }
1192 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
1193 {
1194   \tl_use:N \l__enumext_after_list_args_viii_tl
1195 }

```

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

13.22 Setting keys for multicol 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.

```

1196 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1197 {
1198   \keys_define:nn { enumext / #1 }
1199   {
1200     mini-env .dim_set:c = { l__enumext_minipage_right_#2_dim },
1201     mini-env .value_required:n = true,
1202     mini-sep .dim_set:c = { l__enumext_minipage_hsep_#2_dim },
1203     mini-sep .initial:n = 0.3333em,
1204     mini-sep .value_required:n = true,
1205     columns-sep .dim_set:c = { l__enumext_columns_sep_#2_dim },
1206     columns-sep .value_required:n = true,
1207     columns .int_set:c = { l__enumext_columns_#2_int },
1208     columns .initial:n = 1,
1209     columns .value_required:n = true,
1210   }
1211 }
1212 \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.

```

1213 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1214 {
1215   \keys_define:nn { enumext / #1 }
1216   {
1217     mini-right .tl_gset:c = { g__enumext_miniright_code_#2_tl },
1218     mini-right .value_required:n = true,
1219     mini-right* .code:n = {

```

```

1220 \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
1221 \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1222 },
1223 mini-right* .value_required:n = true,
1224 }
1225 }
1226 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

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

13.23 Adjustment of vertical spaces for multicol

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

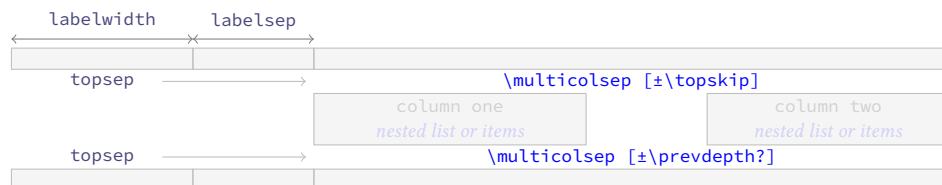


Figure 7: Representation of the vertical space in `multicol` for a nested level.

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

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

13.23.1 Adjustment of vertical spaces for multicol in enumext

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

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

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

```

1227 \cs_new_protected:Nn \__enumext_multi_set_vskip:
1228 {
1229   \skip_set:cn { \l__enumext_multicol_above_ \__enumext_level: _skip }
1230   {
1231     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1232   }
1233   \skip_set:cn { \l__enumext_multicol_below_ \__enumext_level: _skip }
1234   {
1235     \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1236   }
1237   \__enumext_add_pre_parsep:
1238 }

```

(End of definition for `__enumext_multi_set_vskip:`)

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

```

1239 \cs_new_protected:Nn \__enumext_add_pre_parsep:
1240 {
1241   \int_case:nn { \l__enumext_level_int }
1242   {
1243     { 2 }{
1244       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1245       {
1246         \skip_add:Nn \l__enumext_multicol_above_ii_skip
1247         {
1248           \l__enumext_parsep_i_skip
1249         }
1250       }
1251     }

```

```

1252     { 3 }{
1253         \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1254         {
1255             \skip_add:Nn \l__enumext_multicols_above_iii_skip
1256             {
1257                 \l__enumext_parsep_ii_skip
1258             }
1259         }
1260     }
1261     { 4 }{
1262         \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1263         {
1264             \skip_add:Nn \l__enumext_multicols_above_iv_skip
1265             {
1266                 \l__enumext_parsep_iii_skip
1267             }
1268         }
1269     }
1270 }
1271 }

```

(End of definition for `__enumext_add_pre_parsep:`)

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

```

1272 \cs_new_protected:Nn \__enumext_multi_addvspace:
1273 {
1274     \__enumext_multi_set_vskip:
1275     \mode_if_vertical:T
1276     {
1277         \skip_add:cn { \l__enumext_multicols_above_ \__enumext_level: _skip }
1278         {
1279             \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1280         }
1281         \skip_add:cn { \l__enumext_multicols_below_ \__enumext_level: _skip }
1282         {
1283             \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1284         }
1285     }
1286     \par\nopagebreak
1287     \addvspace{ \skip_use:c { \l__enumext_multicols_above_ \__enumext_level: _skip } }
1288 }

```

(End of definition for `__enumext_multi_addvspace:`)

13.23.2 Adjustment of vertical spaces for multicols in keyans

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

`__enumext_keyans_multi_addvspace:`

```

1289 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1290 {
1291     \skip_set:Nn \l__enumext_multicols_above_v_skip
1292     {
1293         \l__enumext_topsep_v_skip
1294     }
1295     \skip_set:Nn \l__enumext_multicols_below_v_skip
1296     {
1297         \l__enumext_topsep_v_skip
1298     }
1299 }
1300 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1301 {
1302     \__enumext_keyans_multi_set_vskip:
1303     \mode_if_vertical:T
1304     {
1305         \skip_add:Nn \l__enumext_multicols_above_v_skip
1306         {
1307             \skip_use:N \l__enumext_partopsep_v_skip
1308         }
1309         \skip_add:Nn \l__enumext_multicols_below_v_skip

```

```

1310     {
1311         \skip_use:N \l__enumext_partopsep_v_skip
1312     }
1313 }
1314 \par\nopagebreak
1315 \addvspace{ \l__enumext_multicols_above_v_skip }
1316 }

```

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

13.24 Adjustment of vertical spaces for minipage

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



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

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

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

- Again I must make clear that the implementation here is a “bit questionable”, but hunting the spaces (glue) produced by the `minipage` environment is quite complicated, even more if `multicols` it is nested. The setting of the values was more “trial and error” (approx to `\strutbox`), using the help of the `lua-visual-debug`[15] package, again my attempts to find the correct values using `\showoutput` and `\showboxdepth` absolutely failed.

13.24.1 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 $\langle vertical mode \rangle$ and we will add `\partopsep`, followed by that we set the value of `\l__enumext_minipage_after_skip`.

```

1317 \cs_new_protected:Nn \__enumext_minipage_set_skip:
1318 {
1319     \skip_set:Nn \l__enumext_minipage_right_skip
1320     {
1321         \skip_use:c { \l__enumext_topsep_ \__enumext_level: _skip }
1322     }
1323     \mode_if_vertical:T
1324     {
1325         \skip_add:Nn \l__enumext_minipage_right_skip
1326         {
1327             \skip_use:c { \l__enumext_partopsep_ \__enumext_level: _skip }
1328         }
1329     }
1330     \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:`.

```

1331     \skip_set_eq:cN
1332     { \l__enumext_multicols_above_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1333     \skip_set_eq:cN
1334     { \l__enumext_multicols_below_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1335     \__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`.

```

1336     \int_compare:nNt
1337     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1338     {
1339         \skip_zero:N \topskip

```



```

1340     \skip_set_eq:Nc \multicolsep { \__enumext_multicols_above_ \__enumext_level: _skip }
1341   }
1342 }

```

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 T_EX is in *horizontal mode* or *vertical mode*. Here we use the plain T_EX 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*.

```

1343 \cs_new_protected:Nn \__enumext_minipage_add_space:
1344 {
1345   \__enumext_minipage_set_skip:
1346   \__enumext_unskip_unkern:
1347   \mode_if_vertical:TF
1348   {
1349     \nopagebreak\nointerlineskip
1350   }
1351   {
1352     \par\nopagebreak\nointerlineskip
1353     \skip_zero:c { \__enumext_partopsep_ \__enumext_level: _skip }
1354   }
1355   \int_compare:nNnTF
1356   { \int_use:c { \__enumext_columns_ \__enumext_level: _int } } > { 1 }
1357   {
1358     \addvspace{ 0.445\box_ht:N \strutbox }
1359   }
1360   {
1361     \addvspace{ 0.250\box_ht:N \strutbox }
1362   }
1363 }

```

(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.

```

1364 \cs_new_protected:Nn \__enumext_pre_itemsep_skip:
1365 {
1366   \int_case:nn { \__enumext_level_int }
1367   {
1368     { 2 }{
1369       \skip_if_eq:nnTF
1370       { \__enumext_itemsep_i_skip } { \__enumext_minipage_after_skip }
1371       {
1372         \skip_set:Nn \__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1373         \skip_set:Nn \__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1374       }
1375       {
1376         \dim_compare:nNnT
1377         { \__enumext_itemsep_i_skip } < { \__enumext_minipage_after_skip }
1378         {
1379           \skip_sub:Nn
1380           \__enumext_minipage_after_skip { \__enumext_itemsep_i_skip }
1381           \skip_sub:Nn
1382           \__enumext_multicols_below_ii_skip { \__enumext_itemsep_i_skip }
1383           \skip_add:Nn
1384           \__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1385           \skip_add:Nn
1386           \__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1387         }
1388         \dim_compare:nNnT
1389         { \__enumext_itemsep_i_skip } > { \__enumext_minipage_after_skip }
1390         {
1391           \skip_set:Nn \__enumext_minipage_temp_skip
1392           {
1393             \__enumext_itemsep_i_skip - \__enumext_minipage_after_skip
1394           }
1395           \skip_sub:Nn
1396           \__enumext_minipage_after_skip { \__enumext_itemsep_i_skip }
1397           \skip_sub:Nn
1398           \__enumext_multicols_below_ii_skip { \__enumext_itemsep_i_skip }

```

```

1399         \skip_add:Nn
1400         \l__enumext_minipage_after_skip
1401         { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1402     \skip_add:Nn
1403     \l__enumext_multicols_below_ii_skip
1404     { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1405 }
1406 }
1407 }
1408 { 3 }{
1409     \skip_if_eq:nnTF
1410     { \l__enumext_itemsep_ii_skip } { \c_zero_skip }
1411     {
1412         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1413         \skip_set:Nn \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1414     }
1415     {
1416         \dim_compare:nNnT
1417         { \l__enumext_itemsep_ii_skip } < { \l__enumext_minipage_after_skip }
1418         {
1419             \skip_sub:Nn
1420             \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1421             \skip_sub:Nn
1422             \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1423             \skip_add:Nn
1424             \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1425             \skip_add:Nn
1426             \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1427         }
1428         \dim_compare:nNnT
1429         { \l__enumext_itemsep_ii_skip } > { \l__enumext_minipage_after_skip }
1430         {
1431             \skip_set:Nn \l__enumext_minipage_temp_skip
1432             {
1433                 \l__enumext_itemsep_ii_skip - \l__enumext_minipage_after_skip
1434             }
1435             \skip_sub:Nn
1436             \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1437             \skip_sub:Nn
1438             \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1439             \skip_add:Nn
1440             \l__enumext_minipage_after_skip
1441             { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1442             \skip_add:Nn
1443             \l__enumext_multicols_below_iii_skip
1444             { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1445         }
1446     }
1447 }
1448 { 4 }{
1449     \skip_if_eq:nnTF { \l__enumext_itemsep_iii_skip } { \c_zero_skip }
1450     {
1451         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1452         \skip_set:Nn \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1453     }
1454     {
1455         \dim_compare:nNnT
1456         { \l__enumext_itemsep_iii_skip } < { \l__enumext_minipage_after_skip }
1457         {
1458             \skip_sub:Nn
1459             \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1460             \skip_sub:Nn
1461             \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1462             \skip_add:Nn
1463             \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1464             \skip_add:Nn
1465             \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1466         }
1467         \dim_compare:nNnT
1468         { \l__enumext_itemsep_iii_skip } > { \l__enumext_minipage_after_skip }
1469         {

```

```

1470         \skip_set:Nn \l__enumext_minipage_temp_skip
1471         {
1472             \l__enumext_itemsep_iii_skip - \l__enumext_minipage_after_skip
1473         }
1474         \skip_sub:Nn
1475             \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1476         \skip_sub:Nn
1477             \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1478         \skip_add:Nn
1479             \l__enumext_minipage_after_skip
1480             { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1481         \skip_add:Nn
1482             \l__enumext_multicols_below_iv_skip
1483             { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1484         }
1485     }
1486 }
1487 }
1488 }

```

(End of definition for `__enumext_pre_itemsep_skip:`)

13.24.2 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`.

```

1489 \cs_new_protected:Nn \__enumext_keyans_minipage_set_skip:
1490 {
1491     \skip_zero:N \l__enumext_minipage_after_skip
1492     \skip_zero:N \l__enumext_minipage_left_skip
1493     \skip_zero:N \l__enumext_minipage_right_skip
1494     \skip_set:Nn \l__enumext_minipage_right_skip
1495     {
1496         \l__enumext_topsep_v_skip
1497     }
1498     \mode_if_vertical:T
1499     {
1500         \skip_add:Nn \l__enumext_minipage_right_skip
1501         {
1502             \l__enumext_partopsep_v_skip
1503         }
1504     }
1505     \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip
1506     \skip_set_eq:NN \l__enumext_multicols_above_v_skip \l__enumext_minipage_right_skip
1507     \skip_set_eq:NN \l__enumext_multicols_below_v_skip \l__enumext_minipage_right_skip
1508     \__enumext_keyans_pre_itemsep_skip:
1509     \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
1510     {
1511         \skip_zero:N \topskip
1512         \skip_set_eq:NN \multicolsep \l__enumext_minipage_right_skip
1513     }
1514 }
1515 \cs_new_protected:Nn \__enumext_keyans_minipage_add_space:
1516 {
1517     \__enumext_keyans_minipage_set_skip:
1518     \__enumext_unskip_unkern:
1519     \mode_if_vertical:TF
1520     {
1521         \nopagebreak\nointerlineskip
1522     }
1523     {
1524         \par\nopagebreak\nointerlineskip
1525         \skip_zero:N \l__enumext_partopsep_v_skip
1526     }
1527     \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1528     {
1529         \addvspace{ 0.445\box_ht:N \strutbox }
1530     }
1531     {
1532         \addvspace{ 0.250\box_ht:N \strutbox }
1533     }

```

```

1534 }
1535 \cs_new_protected:Nn \__enumext_keyans_pre_itemsep_skip:
1536 {
1537   \skip_if_eq:nnTF
1538   { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1539   {
1540     \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1541     \skip_set:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1542   }
1543   {
1544     \dim_compare:nNnT
1545     { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1546     {
1547       \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1548       \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1549       \skip_add:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1550       \skip_add:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1551     }
1552     \dim_compare:nNnT
1553     { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1554     {
1555       \skip_set:Nn \l__enumext_minipage_temp_skip
1556       {
1557         \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1558       }
1559       \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1560       \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1561       \skip_add:Nn \l__enumext_minipage_after_skip
1562       { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1563       \skip_add:Nn \l__enumext_multicols_below_v_skip
1564       { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1565     }
1566   }
1567 }

```

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

13.24.3 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*`.

```

1568 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1569 {
1570   \skip_zero_new:N \l__enumext_minipage_left_skip
1571   \skip_gzero_new:N \g__enumext_minipage_right_skip
1572   \skip_gzero_new:N \g__enumext_minipage_after_skip
1573   \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1574   {
1575     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1576     \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1577   }
1578   {
1579     \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1580     \skip_gset:Nn \g__enumext_minipage_right_skip
1581     {
1582       \l__enumext_topsep_vii_skip
1583     }
1584     \skip_gset:Nn \g__enumext_minipage_after_skip
1585     {
1586       0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1587     }
1588   }
1589 }
1590 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1591 {
1592   \skip_zero_new:N \l__enumext_minipage_after_skip
1593   \skip_zero_new:N \l__enumext_minipage_left_skip
1594   \skip_zero_new:N \l__enumext_minipage_right_skip
1595   \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1596   {

```

```

1597     \skip_set:Nn \l__enumext_minipage_left_skip
1598     {
1599         0.5\box_dp:N \strutbox
1600     }
1601     \skip_set:Nn \l__enumext_minipage_right_skip
1602     {
1603         \l__enumext_partopsep_viii_skip
1604     }
1605     \skip_set:Nn \l__enumext_minipage_after_skip
1606     {
1607         1.6\box_dp:N \strutbox
1608     }
1609 }
1610 {
1611     \skip_set:Nn \l__enumext_minipage_left_skip
1612     {
1613         0.5875\box_dp:N \strutbox
1614     }
1615     \skip_set:Nn \l__enumext_minipage_right_skip
1616     {
1617         \l__enumext_topsep_viii_skip
1618     }
1619     \skip_set:Nn \l__enumext_minipage_after_skip
1620     {
1621         0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1622     }
1623 }
1624 }

```

(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.

```

1625 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1626 {
1627     \__enumext_mini_set_vskip_vii:
1628     \par\nopagebreak
1629     \addvspace { \l__enumext_minipage_left_skip }
1630 }
1631 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1632 {
1633     \__enumext_mini_set_vskip_viii:
1634     \par\nopagebreak
1635     \addvspace { \l__enumext_minipage_left_skip }
1636 }

```

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

13.24.4 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.

```

1637 \NewDocumentCommand \miniright { s }
1638 {
1639     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1640     {
1641         \msg_error:nnn { enumext } { wrong-miniright-place }
1642     }
1643     % outside
1644     \bool_lazy_and:nnT
1645     { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }

```

```

1646     { \int_compare:nNn { \l__enumext_level_h_int } = { 0 } }
1647     {
1648       \msg_error:nnn { enumext } { wrong-miniright-place }
1649     }
1650   % starred env
1651   \bool_lazy_and:nnT
1652     { \bool_if_p:N \g__enumext_starred_bool }
1653     { \bool_not_p:n { \l__enumext_standar_bool } }
1654     {
1655       \msg_error:nnn { enumext } { wrong-miniright-starred }
1656     }
1657   % exec
1658   \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
1659     {
1660       \__enumext_keyans_mini_right_cmd:n {#1}
1661     }
1662     { \__enumext_mini_right_cmd:n {#1} }
1663 }

```

(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.

```

1664 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1665 {
1666   \dim_compare:nNnTF
1667     { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1668     {
1669     \__enumext_multicols_stop:
1670     \int_compare:nNnT
1671       { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } = { 1 }
1672       {
1673         \par\addvspace{ \l__enumext_minipage_after_skip }
1674       }
1675     \end__enumext_mini_page
1676     \hfill
1677     \__enumext_mini_page{ \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } }
1678     \par\nointerlineskip
1679     \addvspace { \l__enumext_minipage_right_skip }
1680     \bool_if:nF {#1}
1681     {
1682       \centering
1683     }
1684     \int_gzero:N \g__enumext_minipage_stat_int
1685   }
1686   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1687 % paranoia
1688 \RenewDocumentCommand \miniright { s }
1689 {
1690   \msg_error:nn { enumext } { many-miniright-used }
1691 }
1692 }

```

(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.

```

1693 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1694 {
1695   \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1696   {
1697     \__enumext_keyans_multicols_stop:
1698     \int_compare:nNnT { \l__enumext_columns_v_int } = { 1 }
1699     {
1700       \par\addvspace{ \l__enumext_minipage_after_skip }
1701     }

```



```

1702     \end__enumext_mini_page
1703     \hfill
1704     \__enumext_mini_page{ \__enumext_minipage_right_v_dim }
1705     \par\nointerlineskip
1706     \addvspace { \__enumext_minipage_right_skip }
1707     \bool_if:nF {#1}
1708     {
1709         \centering
1710     }
1711     \int_gzero:N \g__enumext_minipage_stat_int
1712 }
1713 { \msg_error:nnn { enumext } { wrong-miniright-use } }
1714 % paranoia
1715 \RenewDocumentCommand \miniright { s }
1716 {
1717     \msg_error:nn { enumext } { many-miniright-used }
1718 }
1719 }

```

(End of definition for __enumext_keyans_mini_right_cmd:n.)

13.25 Setting above and below keys

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

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

```

1720 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1721 {
1722     \keys_define:nn { enumext / #1 }
1723     {
1724         above .skip_set:c = { \__enumext_vspace_above_#2_skip },
1725         above .value_required:n = true,
1726         above* .code:n      = \bool_set_true:c { \__enumext_vspace_a_star_#2_bool }
1727             \keys_set:nn { enumext / #1 } { above = {##1} },
1728         above* .value_required:n = true,
1729         below .skip_set:c = { \__enumext_vspace_below_#2_skip },
1730         below .value_required:n = true,
1731         below* .code:n      = \bool_set_true:c { \__enumext_vspace_b_star_#2_bool }
1732             \keys_set:nn { enumext / #1 } { below = {##1} },
1733         below* .value_required:n = true,
1734     }
1735 }
1736 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

13.25.1 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.

```

1737 \cs_new_protected:Nn \__enumext_vspace_above:
1738 {
1739     \skip_if_eq:nnF
1740     { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1741     {
1742         \bool_if:cTF { \__enumext_vspace_a_star_ \__enumext_level: _bool }
1743         {
1744             \vspace*{ \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1745         }
1746         {
1747             \vspace { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1748         }
1749     }
1750 }

```

(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.

```

1751 \cs_new_protected:Nn \__enumext_vspace_below:
1752 {
1753   \skip_if_eq:nnF
1754   { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1755   {
1756     \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1757     {
1758       \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1759     }
1760     {
1761       \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1762     }
1763   }
1764 }

```

(End of definition for `__enumext_vspace_below:`.)

13.25.2 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.

```

1765 \cs_new_protected:Nn \__enumext_vspace_above_v:
1766 {
1767   \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1768   {
1769     \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1770     {
1771       \vspace*{ \l__enumext_vspace_above_v_skip }
1772     }
1773     { \vspace { \l__enumext_vspace_above_v_skip } }
1774   }
1775 }

```

(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.

```

1776 \cs_new_protected:Nn \__enumext_vspace_below_v:
1777 {
1778   \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1779   {
1780     \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1781     {
1782       \vspace*{ \l__enumext_vspace_below_v_skip }
1783     }
1784     { \vspace { \l__enumext_vspace_below_v_skip } }
1785   }
1786 }

```

(End of definition for `__enumext_vspace_below_v:`.)

13.25.3 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:`

```

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

```

```

1802         \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1803         {
1804             \vspace*{ \l__enumext_vspace_above_viii_skip }
1805         }
1806         { \vspace { \l__enumext_vspace_above_viii_skip } }
1807     }
1808 }

```

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

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.

```

1809 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1810 {
1811     \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1812     {
1813         \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1814         {
1815             \vspace*{ \l__enumext_vspace_below_vii_skip }
1816         }
1817         { \vspace { \l__enumext_vspace_below_vii_skip } }
1818     }
1819 }
1820 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1821 {
1822     \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1823     {
1824         \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1825         {
1826             \vspace*{ \l__enumext_vspace_below_viii_skip }
1827         }
1828         { \vspace { \l__enumext_vspace_below_viii_skip } }
1829     }
1830 }

```

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

13.26 Setting series, resume and resume* keys

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

We define the keys `series`, `resume` and `resume*` for the “all levels” of `enumext` and `enumext*`. Here we do not need to make sure that `\printkeyans` is not running otherwise the startup value of the environments would be increased when using `resume` or `resume*` keys.

```

1831 \cs_set_protected:Npn \__enumext_tmp:n #1
1832 {
1833     \keys_define:nn { enumext / #1 }
1834     {
1835         series .str_set:N = \l__enumext_series_name_str,
1836         series .value_required:n = true,
1837         resume* .code:n = {
1838             \bool_if:NF \l__enumext_print_keyans_cmd_bool
1839             {
1840                 \__enumext_resume_star:
1841             }
1842         },
1843         resume* .value_forbidden:n = true,
1844     }
1845 }
1846 \clist_map_inline:nn {level-1, level-2, level-3, level-4, enumext*} { \__enumext_tmp:n {#1} }
1847 \cs_set_protected:Npn \__enumext_tmp:n #1
1848 {
1849     \keys_define:nn { enumext / #1 }
1850     {
1851         resume .code:n = {
1852             \bool_if:NF \l__enumext_print_keyans_cmd_bool
1853             {
1854                 \__enumext_resume_series:n {##1}

```

```

1855         }
1856     },
1857 }
1858 }
1859 \clist_map_inline:nn {level-1, level-2, level-3, level-4} { \__enumext_tmp:n {#1} }
1860 \keys_define:nn { enumext / enumext* }
1861 {
1862     resume .code:n = {
1863         \bool_if:NF \__enumext_print_keyans_cmd_bool
1864         {
1865             \__enumext_resume_series_vii:n {#1}
1866         }
1867     },
1868 }

```

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

13.26.1 Internal functions for series key

The function `__enumext_filter_series:n` will be in charge of filtering the *⟨keys⟩* 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
1869 \cs_new:Npn \__enumext_filter_series:n #1
1870 {
1871     \use:e
1872     {
1873         \keyval_parse:NNn
1874         \__enumext_filter_series_key:n
1875         \__enumext_filter_series_pair:nn {#1}
1876     }
1877 }

```

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

```

1878 \cs_new:Npn \__enumext_filter_series_key:n #1
1879 {
1880     \str_case:nnF {#1}
1881     {
1882         { resume } {} { resume* } {} { base-fix } {}
1883     }
1884     { , { \exp_not:n {#1} } }
1885 }

```

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

```

1886 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1887 {
1888     \str_case:nnF {#1}
1889     {
1890         { series } {} { resume } {} { start } {}
1891         { start* } {} { save-ans } {} { save-key } {}
1892     }
1893     { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1894 }

```

(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 *⟨keys⟩* in the global variable `\g__enumext_series_⟨series name⟩_tl` along with the creation of the integer variable `\g__enumext_series_⟨series name⟩_int` when the key is passed as an argument; otherwise, it will check the state of the boolean variable `\l__enumext_resume_X_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_X_bool` is set to true by the function `__enumext_resume_counter:n` which is used by the keys `resume` and `resume*`, in this case we must make sure it is set to false so that it does not overwrite the default filtered *⟨keys⟩*. This function is passed to the function `__enumext_parse_keys:n` in the `enumext` environment definition (§13.41) and to the function `__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§13.46).

```

1895 \cs_new_protected:Npn \__enumext_parse_series:n #1
1896 {
1897     \str_if_empty:NTF \l__enumext_series_name_str
1898     {
1899         \int_compare:nNnT { \l__enumext_level_int } > { 0 }

```

```

1900         {
1901             \bool_if:cF { \__enumext_resume_ \__enumext_level: _bool }
1902             {
1903                 \__enumext_resume_last:n {#1}
1904             }
1905         }
1906     \int_compare:nNnT { \__enumext_level_h_int } = { 1 }
1907     {
1908         \bool_if:NF \__enumext_resume_vii_bool
1909         {
1910             \__enumext_resume_last:n {#1}
1911         }
1912     }
1913 }
1914 {
1915     \int_compare:nNnT { \__enumext_level_int } > { 0 }
1916     {
1917         \tl_gclear_new:c { g__enumext_series_ \__enumext_series_name_str _ \__enumext_level:
1918         \tl_gset:ce
1919             { g__enumext_series_ \__enumext_series_name_str _ \__enumext_level: _tl }
1920             { \__enumext_filter_series:n {#1} }
1921         \int_if_exist:cF { g__enumext_series_ \__enumext_series_name_str _ \__enumext_level:
1922             {
1923                 \int_new:c { g__enumext_series_ \__enumext_series_name_str _ \__enumext_level: _
1924             }
1925         }
1926     \int_compare:nNnT { \__enumext_level_h_int } = { 1 }
1927     {
1928         \tl_gclear_new:c { g__enumext_series_ \__enumext_series_name_str _vii_tl }
1929         \tl_gset:ce
1930             { g__enumext_series_ \__enumext_series_name_str _vii_tl }
1931             { \__enumext_filter_series:n {#1} }
1932         \int_if_exist:cF { g__enumext_series_ \__enumext_series_name_str _vii_int }
1933             {
1934                 \int_new:c { g__enumext_series_ \__enumext_series_name_str _vii_int }
1935             }
1936     }
1937 }
1938 }

```

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_series_X_tl` for the *enumext* environment and in the variable `\g__enumext_series_vii_tl` for the *enumext** environment.

```

1939 \cs_new_protected:Npn \__enumext_resume_last:n #1
1940 {
1941     \int_compare:nNnT { \__enumext_level_int } > { 0 }
1942     {
1943         \tl_gclear:c { g__enumext_not_key_series_ \__enumext_level: _tl }
1944         \tl_gset:ce
1945             { g__enumext_not_key_series_ \__enumext_level: _tl } { \__enumext_filter_series:n {#1} }
1946     }
1947     \int_compare:nNnT { \__enumext_level_h_int } = { 1 }
1948     {
1949         \tl_gclear:N \g__enumext_not_key_series_vii_tl
1950         \tl_gset:Ne \g__enumext_not_key_series_vii_tl { \__enumext_filter_series:n {#1} }
1951     }
1952 }

```

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

13.26.2 Internal function to save counter value

```

\__enumext_standar_save_counter:
\__enumext_standar_save_counter_aux:
\__enumext_starred_save_counter:
\__enumext_starred_save_counter_aux:

```

The `__enumext_standar_save_counter:` and `__enumext_starred_save_counter:` functions 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_X_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_name_str` and `\l__enumext_series_name_tl` contain the same *{⟨series name⟩}* but are executed at different moments, the integer variable with `\l__enumext_series_name_str` sets the value when execute *series*=*{⟨series name⟩}* and the integer variable with `\l__enumext_series_name_tl` sets the subsequent

values when use `resume={(series name)}`. This function is passed to the `enumext` environment definition (§13.41) and the `enumext*` environment definition (§13.46).

```

1953 \cs_new_protected:Nn \__enumext_standar_save_counter:
1954 {
1955     \bool_if:NTF \g__enumext_standar_bool % enumext NOT nested
1956     {
1957         \__enumext_standar_save_counter_aux:
1958         \int_compare:nNtT { \l__enumext_level_int } = { 1 }
1959         {
1960             \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1961             {
1962                 \int_gset_eq:cN
1963                 { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1964             }
1965         }
1966     }
1967     {
1968         \__enumext_standar_save_counter_aux:
1969     }
1970 }
1971 \cs_new_protected:Nn \__enumext_standar_save_counter_aux:
1972 {
1973     \str_if_empty:NF \l__enumext_series_name_str
1974     {
1975         \int_gset_eq:cc
1976         { g__enumext_series_ \l__enumext_series_name_str _ \__enumext_level: _int } { c@enumX \__enumext_level: }
1977     }
1978     \tl_if_empty:NTF \l__enumext_series_name_tl
1979     {
1980         \str_if_empty:NT \l__enumext_series_name_str
1981         {
1982             \tl_if_empty:NT \l__enumext_store_name_tl
1983             {
1984                 \int_gset_eq:cc
1985                 { g__enumext_resume_ \__enumext_level: _int } { c@enumX \__enumext_level: }
1986             }
1987         }
1988     }
1989     {
1990         \int_if_exist:cT { g__enumext_series_ \l__enumext_series_name_tl _ \__enumext_level: _int }
1991         {
1992             \int_gset_eq:cc
1993             { g__enumext_series_ \l__enumext_series_name_tl _ \__enumext_level: _int } { c@enumX \__enumext_level: }
1994         }
1995     }
1996 }
1997 \cs_new_protected:Nn \__enumext_starred_save_counter:
1998 {
1999     \bool_if:NTF \g__enumext_starred_bool % enumext* NOT nested
2000     {
2001         \__enumext_starred_save_counter_aux:
2002         \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
2003         {
2004             \int_gset_eq:cN
2005             { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
2006         }
2007     }
2008     {
2009         \__enumext_starred_save_counter_aux:
2010     }
2011 }
2012 \cs_new_protected:Nn \__enumext_starred_save_counter_aux:
2013 {
2014     \str_if_empty:NF \l__enumext_series_name_str
2015     {
2016         \int_gset_eq:cN
2017         { g__enumext_series_ \l__enumext_series_name_str _vii_int } \value{enumXvii}
2018     }
2019     \tl_if_empty:NTF \l__enumext_series_name_tl
2020     {
2021         \str_if_empty:NT \l__enumext_series_name_str

```

```

2022         {
2023             \tl_if_empty:NT \l__enumext_store_name_tl
2024             {
2025                 \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
2026             }
2027         }
2028     }
2029     {
2030         \int_if_exist:cT { g__enumext_series_ \l__enumext_series_name_tl _vii_int }
2031         {
2032             \int_gset_eq:cN
2033             { g__enumext_series_ \l__enumext_series_name_tl _vii_int } \value{enumXvii}
2034         }
2035     }
2036 }

```

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

13.26.3 Internal functions for resume key

```

\__enumext_resume_series:n
\__enumext_resume_series_vii:n

```

The functions `__enumext_resume_series:n` and `__enumext_resume_series_vii: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:n` 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.

```

2037 \cs_new_protected:Npn \__enumext_resume_series:n #1
2038 {
2039     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2040     {
2041         \tl_if_empty:nTF {#1}
2042         {
2043             \__enumext_resume_counter:n { }
2044         }
2045         {
2046             \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _ \__enumext_level: _tl }
2047             {
2048                 \__enumext_resume_counter:n {#1}
2049                 \exp_args:Ne \keys_set:nv { enumext / level-\int_use:N \l__enumext_level_int }
2050                 { g__enumext_series_ \tl_to_str:n {#1} _ \__enumext_level: _tl }
2051             }
2052             {
2053                 \msg_error:nnn { enumext } { unknown-series-standar } {#1}
2054             }
2055         }
2056     }
2057 }
2058 \cs_new_protected:Npn \__enumext_resume_series_vii:n #1
2059 {
2060     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
2061     {
2062         \tl_if_empty:nTF {#1}
2063         {
2064             \__enumext_resume_counter:n { }
2065         }
2066         {
2067             \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _vii_tl }
2068             {
2069                 \__enumext_resume_counter:n {#1}
2070                 \keys_set:nv { enumext / enumext* }
2071                 { g__enumext_series_ \tl_to_str:n {#1} _vii_tl }
2072             }
2073             {
2074                 \msg_error:nnn { enumext } { unknown-series-starred } {#1}
2075             }
2076         }
2077     }
2078 }

```

(End of definition for `__enumext_resume_series:n` and `__enumext_resume_series_vii:n`)

```

\__enumext_resume_counter:n
\__enumext_resume_counter:
\__enumext_resume_counter_series:

```

The function `__enumext_resume_counter:n` will set the variable `\l__enumext_resume_X_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`:

```

2079 \cs_new_protected:Npn \__enumext_resume_counter:n #1
2080 {
2081   \int_compare:nNtT { \l__enumext_level_int } > { 0 }
2082   {
2083     \bool_set_true:c { l__enumext_resume_ \__enumext_level: _bool }
2084     \tl_clear:N \l__enumext_series_name_tl
2085     \tl_set:Nn \l__enumext_series_name_tl {#1}
2086     \tl_if_empty:NTF \l__enumext_series_name_tl
2087     {
2088       \__enumext_resume_counter:
2089     }
2090     {
2091       \__enumext_resume_counter_series:
2092     }
2093   }
2094   \int_compare:nNtT { \l__enumext_level_h_int } = { 1 }
2095   {
2096     \bool_set_true:N \l__enumext_resume_vii_bool
2097     \tl_clear:N \l__enumext_series_name_tl
2098     \tl_set:Nn \l__enumext_series_name_tl {#1}
2099     \tl_if_empty:NTF \l__enumext_series_name_tl
2100     {
2101       \__enumext_resume_counter:
2102     }
2103     {
2104       \__enumext_resume_counter_series:
2105     }
2106   }
2107 }

```

The `__enumext_resume_counter`: function is executed when the `resume` key is used *without value*, only the counters for the “levels” of the environments will be set. If the `save-ans` key is active it will set the counter according to the value of the integer variable created by that key.

```

2108 \cs_new_protected:Nn \__enumext_resume_counter:
2109 {
2110   \cs_set:Npn \__enumext_tmp:n ##1
2111   {
2112     \exp_args:Ne \int_set:cn { l__enumext_start_ \int_to_roman:n {##1} _int }
2113     {
2114       \int_use:c { g__enumext_resume_ \int_to_roman:n {##1} _int } + 1
2115     }
2116   }
2117   \int_compare:nNtT { \l__enumext_level_int } > { 0 }
2118   {
2119     \bool_lazy_and:nnTF
2120     { \bool_if_p:N \l__enumext_standar_first_bool }
2121     { \bool_if_p:N \l__enumext_store_active_bool }
2122     {
2123       \int_set:Nn \l__enumext_start_i_int
2124       {
2125         \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
2126       }
2127       \int_step_function:nnN { 2 } { \l__enumext_level_int } \l__enumext_tmp:n
2128     }
2129     {
2130       \int_step_function:nnN { 1 } { \l__enumext_level_int } \l__enumext_tmp:n
2131     }
2132   }
2133   \int_compare:nNtT { \l__enumext_level_h_int } = { 1 }
2134   {
2135     \bool_lazy_and:nnTF
2136     { \bool_if_p:N \l__enumext_starred_first_bool }
2137     { \bool_if_p:N \l__enumext_store_active_bool }
2138     {
2139       \int_set:Nn \l__enumext_start_vii_int
2140       {
2141         \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
2142       }

```

```

2143     }
2144     {
2145         \int_set:Nn \l__enumext_start_vii_int { \g__enumext_resume_vii_int + 1 }
2146     }
2147 }
2148 }

```

The function `__enumext_resume_counter_series:` will be executed when the `resume={⟨series name⟩}` key is active, setting the counters for the “current level” of the environments according to the value of the integer variables created by the `series` key. If the `save-ans` key is active it will set the counter according to the value of the integer variable created by that key.

```

2149 \cs_new_protected:Nn \__enumext_resume_counter_series:
2150 {
2151     \cs_set:Npn \__enumext_tmp:n ##1
2152     {
2153         \int_if_exist:cT { g__enumext_series_ \l__enumext_series_name_tl _ \int_to_roman:n {##1} }
2154         {
2155             \exp_args:Ne \int_set:cn { l__enumext_start_ \int_to_roman:n {##1} _int }
2156             {
2157                 \int_use:c { g__enumext_series_ \l__enumext_series_name_tl _ \int_to_roman:n {##1} }
2158             }
2159         }
2160     }
2161     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2162     {
2163         \bool_lazy_and:nnTF
2164         { \bool_if_p:N \l__enumext_standar_first_bool }
2165         { \bool_if_p:N \l__enumext_store_active_bool }
2166         {
2167             \int_set:Nn \l__enumext_start_i_int
2168             {
2169                 \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
2170             }
2171             \int_step_function:nnN { 2 } { \l__enumext_level_int } \l__enumext_tmp:n
2172         }
2173         {
2174             \int_step_function:nnN { 1 } { \l__enumext_level_int } \l__enumext_tmp:n
2175         }
2176     }
2177     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
2178     {
2179         \bool_lazy_and:nnTF
2180         { \bool_if_p:N \l__enumext_starred_first_bool }
2181         { \bool_if_p:N \l__enumext_store_active_bool }
2182         {
2183             \int_set:Nn \l__enumext_start_vii_int
2184             {
2185                 \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
2186             }
2187         }
2188         {
2189             \int_set:Nn \l__enumext_start_vii_int
2190             {
2191                 \int_use:c { g__enumext_series_ \l__enumext_series_name_tl _vii_int } + 1
2192             }
2193         }
2194     }
2195 }

```

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

13.26.4 Internal function for resume* key

`__enumext_resume_star:`

The function `__enumext_resume_star:` 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 and `⟨keys⟩` 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.

```

2196 \cs_new_protected:Nn \__enumext_resume_star:
2197 {
2198     \cs_set:Npn \__enumext_tmp:n ##1
2199     {

```

```

2200         \tl_if_empty:cF { g__enumext_not_key_series_ \int_to_roman:n {##1} _tl }
2201         {
2202             \__enumext_resume_counter:n { }
2203             \exp_args:Ne \keys_set:nv
2204             { enumext / level-\int_use:N \__enumext_level_int }
2205             { g__enumext_not_key_series_ \int_to_roman:n {##1} _tl }
2206         }
2207     }
2208     \int_compare:nNnT { \__enumext_level_int } > { 0 }
2209     {
2210         \int_step_function:nnN { 1 } { \__enumext_level_int } \__enumext_tmp:n
2211     }
2212     \int_compare:nNnT { \__enumext_level_h_int } = { 1 }
2213     {
2214         \tl_if_empty:NF \g__enumext_not_key_series_vii_tl
2215         {
2216             \__enumext_resume_counter:n { }
2217             \keys_set:nV { enumext / enumext* } \g__enumext_not_key_series_vii_tl
2218         }
2219     }
2220 }

```

(End of definition for `__enumext_resume_star:.`)

13.27 The `\resetenumext`, `reset` and `reset*` keys

```

2221 \NewDocumentCommand \resetenumext { s O{enumext,1} m }
2222 {
2223     \clist_set:Nn \__enumext_reset_args_clist {#2}
2224     \int_compare:nTF { \clist_count:N \__enumext_reset_args_clist > 2 }
2225     {
2226         \msg_error:nne { enumext-reset } { too-many-elements } {#2}
2227     }
2228     \bool_if:nTF {#1}
2229     {
2230         \__enumext_reset_count_resume_all:n { #3 }
2231     }
2232     {
2233         \int_compare:nTF { \clist_count:N \__enumext_reset_args_clist = 1 }
2234         {
2235             \__enumext_starred_reset:n {#3}
2236         }
2237         {
2238             \int_compare:nTF { \clist_count:N \__enumext_reset_args_clist = 2 }
2239             {
2240                 \__enumext_standar_reset:n {#3}
2241             }
2242             {
2243                 \msg_error:nnn { enumext-reset } { invalid-clist } {#2}
2244             }
2245         }
2246     }
2247 }
2248 % #1 : internal list ; #2 : external (section/chapter/etc...)
2249 \cs_new_protected:Npn \__enumext_reset_count_resume:nn #1 #2
2250 {
2251     \counterwithin* {enumX#1}{#2}
2252     \int_gzero:c { g__enumext_resume_ #1 _int }
2253 }
2254 \cs_new_protected:Npn \__enumext_reset_count_resume_all:n #1
2255 {
2256     \clist_map_inline:nn { i,ii,iii,iv,vii }
2257     {
2258         \__enumext_reset_count_resume:nn { ##1 } { #1 }
2259     }
2260 }
2261 \cs_new_protected:Npn \__enumext_starred_reset:n #1
2262 {
2263     \str_set:Ne \__enumext_reset_name_str { \clist_item:Nn \__enumext_reset_args_clist {1} }
2264     \str_if_eq:eeTF { \__enumext_reset_name_str } { enumext* }
2265     {
2266         \__enumext_reset_count_resume:nn {vii} {#1}

```

```

2267     }
2268     {
2269         \msg_error:nn { enumext-reset } { invalid-single-arg-no-star }
2270     }
2271 }
2272 % standar reset
2273 \cs_new_protected:Npn \__enumext_standar_reset:n #1
2274 {
2275     \str_set:Nx \__enumext_reset_name_str { \clist_item:Nn \__enumext_reset_args_clist {1} }
2276     \int_set:Nn \__enumext_reset_level_int { \clist_item:Nn \__enumext_reset_args_clist {2} }
2277     \str_if_eq:eeTF { \__enumext_reset_name_str } { enumext }
2278     {
2279         \int_case:nnF { \__enumext_reset_level_int }
2280         {
2281             { 1 }
2282             {
2283                 \__enumext_reset_count_resume:nn
2284                 { \int_to_roman:n { \__enumext_reset_level_int } } {#1}
2285             }
2286             { 2 }
2287             {
2288                 \__enumext_reset_count_resume:nn
2289                 { \int_to_roman:n { \__enumext_reset_level_int } } {#1}
2290             }
2291             { 3 }
2292             {
2293                 \__enumext_reset_count_resume:nn
2294                 { \int_to_roman:n { \__enumext_reset_level_int } } {#1}
2295             }
2296             { 4 }
2297             {
2298                 \__enumext_reset_count_resume:nn
2299                 { \int_to_roman:n { \__enumext_reset_level_int } } {#1}
2300             }
2301         }
2302         {
2303             \msg_error:nne { enumext-reset } { out-of-range } { \__enumext_reset_level_int }
2304         }
2305     }
2306     {
2307         \msg_error:nne { enumext-reset } { invalid-package } { \__enumext_reset_name_str }
2308     }
2309 }

```

13.28 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.

13.28.1 Setting save-ans key

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

```

2310 \cs_set_protected:Npn \__enumext_tmp:n #1
2311 {
2312     \keys_define:nn { enumext / #1 }
2313     {
2314         save-ans .code:n = \__enumext_storing_set:n {##1},
2315         save-ans .value_required:n = true,
2316     }
2317 }
2318 \clist_map_inline:nn { level-1, enumext* } { { \__enumext_tmp:n {#1} } }

```

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

13.28.2 Internal functions for save-ans key

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

```

2319 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
2320 {

```

```

2321 \msg_term:nnVV { enumext } { save-ans-log }
2322 \g__enumext_envir_name_tl \l__enumext_store_name_tl
2323 }
2324 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
2325 {
2326 \msg_term:nnVV { enumext } { save-ans-log-hook }
2327 \g__enumext_envir_name_tl \g__enumext_store_name_tl
2328 }

```

(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 $\langle store\ name \rangle$ 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.

```

2329 \cs_new_protected:Npn \__enumext_storing_set:n #1
2330 {
2331 \tl_set:Nx \l__enumext_store_name_tl {#1}
2332 \tl_if_empty:NTF \l__enumext_store_name_tl
2333 {
2334 \bool_lazy_or:nnT
2335 { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
2336 {
2337 \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
2338 }
2339 }
2340 {
2341 \bool_lazy_or:nnT
2342 { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
2343 {
2344 \__enumext_start_save_ans_msg:
2345 \__enumext_storing_exec:
2346 }
2347 }
2348 }

```

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 $\langle store\ name \rangle$ into the variable \g__enumext_store_name_tl.

```

2349 \cs_new_protected:Nn \__enumext_storing_exec:
2350 {
2351 \bool_set_true:N \l__enumext_store_active_bool
2352 \bool_set_true:N \l__enumext_check_answers_bool
2353 \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl

```

The *prop list* \g__enumext_series_ $\langle store\ name \rangle$ _prop and the *sequence* \g__enumext_series_ $\langle store\ name \rangle$ _seq will be created globally to “store content” in case they do not exist together with the integer variable \g__enumext_series_ $\langle store\ name \rangle$ _int used by the keys `resume` and `resume*`.

```

2354 \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
2355 {
2356 \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
2357 \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
2358 }
2359 \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
2360 {
2361 \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
2362 \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
2363 }
2364 \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
2365 {
2366 \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
2367 \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
2368 }
2369 }

```

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

13.28.3 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.

13.28.4 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 2370 \cs_set_protected:Npn \__enumext_tmp:n #1
no-store 2371 {
2372   \keys_define:nn { enumext / #1 }
2373   {
2374     check-ans .bool_set:N = \__enumext_check_ans_key_bool,
2375     check-ans .initial:n = false,
2376     check-ans .value_required:n = true,
2377     no-store .code:n = {
2378       \bool_set_false:N \__enumext_check_answers_bool
2379       \bool_set_false:N \__enumext_check_ans_key_bool
2380     },
2381     no-store .value_forbidden:n = true,
2382   }
2383 }
2384 \clist_map_inline:nn
2385 {
2386   level-1, level-2, level-3, level-4, enumext*
2387 }
2388 { \__enumext_tmp:n {#1} }
```

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

13.28.5 Set-up check answer mechanism

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

```

2389 \cs_new_protected:Nn \__enumext_check_ans_active:
2390 {
2391   \tl_if_empty:NF \l__enumext_store_name_tl
2392   {
2393     \bool_if:NT \l__enumext_check_answers_bool
2394     {
2395       \__enumext_check_ans_level:
2396     }
2397   }
2398 }
```

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”.

```

2399 \cs_new_protected:Nn \__enumext_check_ans_level:
2400 {
2401   \int_case:nn { \l__enumext_level_int }
2402   {
2403     { 1 }{
2404       \bool_lazy_all:nT
2405       {
2406         { \bool_if_p:N \g__enumext_starred_bool }
2407         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
2408       }
2409       {
2410         \int_gdecr:N \g__enumext_item_number_int
2411         \bool_set_false:N \l__enumext_item_number_bool
2412       }
2413     }
2414     { 2 }{
2415       \int_gdecr:N \g__enumext_item_number_int
2416       \bool_set_false:N \l__enumext_item_number_bool
2417     }
2418     { 3 }{
2419       \int_gdecr:N \g__enumext_item_number_int
2420       \bool_set_false:N \l__enumext_item_number_bool
2421     }
2422     { 4 }{
2423       \int_gdecr:N \g__enumext_item_number_int
2424       \bool_set_false:N \l__enumext_item_number_bool
2425     }
2426   }

```

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.

```

2427   \int_case:nn { \l__enumext_level_h_int }
2428   {
2429     { 1 }{
2430       \bool_lazy_all:nT
2431       {
2432         { \bool_if_p:N \g__enumext_standar_bool }
2433         { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
2434       }
2435       {
2436         \int_gdecr:N \g__enumext_item_number_int
2437         \bool_set_false:N \l__enumext_item_number_bool
2438       }
2439     }
2440   }
2441 }

```

(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.

```

2442 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
2443 {
2444   \bool_lazy_and:nnT
2445   { \bool_if_p:N \l__enumext_check_ans_key_bool }
2446   { \bool_if_p:N \g__enumext_standar_bool }
2447   {
2448     \bool_gset_true:N \g__enumext_check_ans_key_bool
2449   }
2450   \bool_lazy_and:nnT
2451   { \bool_if_p:N \l__enumext_check_ans_key_bool }
2452   { \bool_if_p:N \g__enumext_starred_bool }
2453   {
2454     \bool_gset_true:N \g__enumext_check_ans_key_bool
2455   }
2456 }

```


(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:`.

```
2457 \cs_new_protected:Nn \__enumext_item_answer_diff:
2458 {
2459     \int_gset:Nn \g__enumext_item_answer_diff_int
2460     {
2461         \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
2462     }
2463 }
```

(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:`.

```
2464 \cs_new_protected:Nn \__enumext_check_ans_show:
2465 {
2466     \int_case:nn { \g__enumext_item_answer_diff_int }
2467     {
2468         { -1 } { \__enumext_check_ans_msg_less: }
2469         { 0 } { \__enumext_check_ans_msg_same_ok: }
2470         { 1 } { \__enumext_check_ans_msg_greater: }
2471     }
2472 }
2473 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
2474 {
2475     \msg_warning:nnee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2476     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2477 }
2478 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
2479 {
2480     \msg_term:nnee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2481     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2482 }
2483 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
2484 {
2485     \msg_warning:nnee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2486     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2487 }
```

(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:`.

```
2488 \cs_new_protected:Nn \__enumext_check_ans_log:
2489 {
2490     \int_case:nn { \g__enumext_item_answer_diff_int }
2491     {
2492         { -1 } { \__enumext_check_ans_log_msg_less: }
2493         { 0 } { \__enumext_check_ans_log_msg_same_ok: }
2494         { 1 } { \__enumext_check_ans_log_msg_greater: }
2495     }
2496 }
2497 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
2498 {
2499     \msg_log:nnee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2500     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2501 }
2502 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
2503 {
2504     \msg_log:nnee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2505     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2506 }
```

```

2507 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
2508 {
2509     \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2510     { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2511 }

```

(End of definition for __enumext_check_ans_log: and others.)

13.28.6 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.

```

2512 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
2513 {
2514     \int_compare:nNt
2515     { \g__enumext_check_starred_cmd_int } = { 0 }
2516     {
2517         \msg_warning:nnnV
2518         { enumext } { missing-starred }{ #1 } \l__enumext_check_start_line_env_tl
2519     }
2520     \int_compare:nNt
2521     { \g__enumext_check_starred_cmd_int } > { 1 }
2522     {
2523         \msg_warning:nnnV
2524         { enumext } { many-starred }{ #1 } \l__enumext_check_start_line_env_tl
2525     }
2526     \int_gzero:N \g__enumext_check_starred_cmd_int
2527     \tl_clear:N \l__enumext_check_start_line_env_tl
2528 }

```

(End of definition for __enumext_check_starred_cmd:n.)

13.29 Keys and functions associated with storage

13.29.1 Keys for marks, wrap and show

The `enumext` package provides a set of *(keys)* for manipulating “symbol marks” associated with “answers” and how they are displayed and stored in the *sequence* and *prop list* as well as an internal “label and ref” system.

mark-ans*
mark-pos*
mark-sep*
wrap-ans*
wrap-opt
save-sep
show-ans
show-pos

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

```

2529 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2530 {
2531     \keys_define:nn { enumext / #1 }
2532     {
2533         mark-ans* .tl_set:c = { \l__enumext_mark_answer_sym_#2_tl },
2534         mark-ans* .initial:n = \textasteriskcentered,
2535         mark-ans* .value_required:n = true,
2536         mark-pos* .choice:,
2537         mark-pos* / left .code:n = \str_set:cn { \l__enumext_mark_position_#2_str } { l },
2538         mark-pos* / right .code:n = \str_set:cn { \l__enumext_mark_position_#2_str } { r },
2539         mark-pos* / center .code:n = \str_set:cn { \l__enumext_mark_position_#2_str } { c },
2540         mark-pos* / unknown .code:n =
2541             \msg_error:nneee { enumext } { unknown-choice }
2542             { mark-pos } { left,~right,~center } { \exp_not:n {##1} },
2543         mark-pos* .initial:n = right,
2544         mark-pos* .value_required:n = true,
2545         mark-sep* .dim_set:c = { \l__enumext_mark_sym_sep_#2_dim },
2546         mark-sep* .value_required:n = true,
2547         wrap-ans* .cs_set_protected:cp = { __enumext_keyans_wrapper_item_#2:n } ##1,
2548         wrap-ans* .value_required:n = true,
2549         wrap-opt .cs_set_protected:cp = { __enumext_keyans_wrapper_opt_#2:n } ##1,
2550         wrap-opt .initial:n = [{##1}],
2551         wrap-opt .value_required:n = true,
2552         save-sep .tl_set:c = { \l__enumext_store_keyans_item_opt_sep_#2_tl },
2553         save-sep .initial:n = {,~},
2554         save-sep .value_required:n = true,
2555         show-ans .bool_set:N = \l__enumext_show_answer_bool,
2556         show-ans .initial:n = false,
2557         show-ans .value_required:n = true,
2558         show-pos .bool_set:N = \l__enumext_show_position_bool,
2559         show-pos .initial:n = false,

```

```

2560         show-pos .value_required:n = true,
2561     }
2562 }
2563 \clist_map_inline:nn { {keyans}{v}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

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

We add the $\langle keys \rangle$ mark-ref and save-ref related to the “storage system” and internal mechanism of “label and ref” along with the $\langle keys \rangle$ show-ans, show-pos and the $\langle keys \rangle$ mark-ans, mark-pos, mark-sep and wrap-ans for the command `\anskey`, the environment `anskey*` and the the $\langle keys \rangle$ for environments `keyans` and `keyans*` only at the *first level* of `enumext` and `enumext*`.

```

2564 \cs_set_protected:Npn \__enumext_tmp:n #1
2565 {
2566     \keys_define:nn { enumext / #1 }
2567     {
2568         mark-ref .tl_set:N = \__enumext_mark_ref_sym_tl,
2569         mark-ref .initial:n = \textreferencemark,
2570         mark-ref .value_required:n = true,
2571         save-ref .bool_set:N = \__enumext_store_ref_key_bool,
2572         save-ref .initial:n = false,
2573         save-ref .value_required:n = true,
2574         show-ans .bool_set:N = \__enumext_show_answer_bool,
2575         show-ans .initial:n = false,
2576         show-ans .value_required:n = true,
2577         show-pos .bool_set:N = \__enumext_show_position_bool,
2578         show-pos .initial:n = false,
2579         show-pos .value_required:n = true,
2580         mark-ans .tl_set:N = \__enumext_mark_answer_sym_tl,
2581         mark-ans .initial:n = \textasteriskcentered,
2582         mark-ans .value_required:n = true,
2583         mark-sep .dim_set:N = \__enumext_mark_sym_sep_dim,
2584         mark-sep .value_required:n = true,
2585         mark-pos .choice:,
2586         mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
2587         mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
2588         mark-pos / center .code:n = \str_set:Nn \__enumext_mark_position_str { c },
2589         mark-pos / unknown .code:n =
2590             \msg_error:nnee { enumext } { unknown-choice }
2591             { mark-pos } { left,~right,~center } { \exp_not:n {##1} },
2592         mark-pos .initial:n = right,
2593         mark-pos .value_required:n = true,
2594
2595         wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
2596         wrap-ans .initial:n =
2597             {
2598                 \fbox{\parbox[t]{\dimeval{\itemwidth -2\fboxsep -2\fboxrule}}{##1}}
2599             },
2600         wrap-ans .value_required:n = true,
2601         mark-ans* .code:n = {
2602             \keys_set:nn { enumext / keyans } { mark-ans* = {##1} }
2603             \keys_set:nn { enumext / keyans* } { mark-ans* = {##1} }
2604         },
2605         mark-ans* .value_required:n = true,
2606         mark-pos* .code:n = {
2607             \keys_set:nn { enumext / keyans } { mark-pos* = {##1} }
2608             \keys_set:nn { enumext / keyans* } { mark-pos* = {##1} }
2609         },
2610         mark-pos* .value_required:n = true,
2611         mark-sep* .code:n = {
2612             \keys_set:nn { enumext / keyans } { mark-sep* = {##1} }
2613             \keys_set:nn { enumext / keyans* } { mark-sep* = {##1} }
2614         },
2615         mark-sep* .value_required:n = true,
2616         wrap-ans* .code:n = {
2617             \keys_set:nn { enumext / keyans } { wrap-ans* = {##1} }
2618             \keys_set:nn { enumext / keyans* } { wrap-ans* = {##1} }
2619         },
2620         wrap-ans* .value_required:n = true,
2621         wrap-opt .code:n = {
2622             \keys_set:nn { enumext / keyans } { wrap-opt = {##1} }
2623             \keys_set:nn { enumext / keyans* } { wrap-opt = {##1} }

```

```

2624         },
2625         wrap-opt .value_required:n = true,
2626         save-sep .code:n = {
2627             \keys_set:nn { enumext / keyans } { save-sep = {##1} }
2628             \keys_set:nn { enumext / keyans* } { save-sep = {##1} }
2629         },
2630         save-sep .value_required:n = true,
2631     }
2632 }
2633 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

(End of definition for *mark-ref* and others.)

13.29.2 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*.

```

2634 \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2635 {
2636     \bool_if:cF { \__enumext_store_save_key_ \__enumext_level: _bool }
2637     {
2638         \tl_clear:c { \__enumext_store_save_key_ \__enumext_level: _tl }
2639         \tl_set:ce
2640             { \__enumext_store_save_key_ \__enumext_level: _tl }
2641             { \__enumext_filter_save_key:n {#1} }
2642     }
2643 }
2644 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2645 {
2646     \bool_if:NF \__enumext_store_save_key_vii_bool
2647     {
2648         \tl_clear:N \__enumext_store_save_key_vii_tl
2649         \tl_set:Ne \__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2650     }
2651 }

```

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

13.29.3 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.

```

2652 \cs_set_protected:Npn \__enumext_tmp:n #1
2653 {
2654     \keys_define:nn { enumext / enumext* }
2655     {
2656         save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2657         save-key .value_required:n = true,
2658     }
2659     \keys_define:nn { enumext / #1 }
2660     {
2661         save-key .code:n = \__enumext_parse_save_key:n {##1},
2662         save-key .value_required:n = true,
2663     }
2664 }
2665 \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 `l__enumext_store_save_key_X_tl` for `enumext` and `enumext*`.

```

2666 \cs_new_protected:Npn __enumext_parse_save_key:n #1
2667 {
2668   \bool_set_true:c { l__enumext_store_save_key_ __enumext_level: _bool }
2669   \tl_clear:c { l__enumext_save_key_ __enumext_level: _tl }
2670   \tl_set:ce
2671     { l__enumext_store_save_key_ __enumext_level: _tl }
2672     { __enumext_filter_save_key:n {#1} }
2673 }
2674 \cs_new_protected:Npn __enumext_parse_save_key_vii:n #1
2675 {
2676   \bool_set_true:N \l__enumext_store_save_key_vii_bool
2677   \tl_clear:N \l__enumext_store_save_key_vii_tl
2678   \tl_set:Ne \l__enumext_store_save_key_vii_tl { __enumext_filter_save_key:n {#1} }
2679 }

```

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

13.29.4 Internal functions to store optional arguments

```

__enumext_filter_save_key:n
  __enumext_filter_save_key_key:n
  __enumext_filter_save_key_pair:nn

```

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.

```

2680 \cs_new:Npn __enumext_filter_save_key:n #1
2681 {
2682   \use:e
2683     {
2684       \keyval_parse:NNn
2685         __enumext_filter_save_key_key:n
2686         __enumext_filter_save_key_pair:nn {#1}
2687     }
2688 }

```

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.

```

2689 \cs_new:Npn __enumext_filter_save_key_key:n #1
2690 {
2691   \str_case:nnF {#1}
2692     {
2693       { resume } {} { resume* } {} { no-store } {} { base-fix } {}
2694     }
2695     { , { \exp_not:n {#1} } }
2696 }

```

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`, `mark-ans`, `mark-pos`, `mark-sep`, `wrap-ans`, `mark-ans*`, `mark-pos*`, `mark-sep*`, `wrap-ans*`, `wrap-opt`, `save-sep`, `mark-ref`, `mini-env`, `mini-sep`, `mini-right` and `mini-right*` keys.

```

2697 \cs_new:Npn __enumext_filter_save_key_pair:nn #1#2
2698 {
2699   \str_case:nnF {#1}
2700     {
2701       { series } {} { resume } {} { save-ans } {} { save-ref } {}
2702       { save-key } {} { check-ans } {} { show-ans } {} { show-pos } {}
2703       { mark-ans } {} { mark-pos } {} { mark-sep } {} { wrap-ans } {}
2704       { mark-ans* } {} { mark-pos* } {} { mark-sep* } {} { wrap-ans* } {}
2705       { wrap-opt } {} { save-sep } {} { mark-ref } {} { mini-env } {}
2706       { mini-sep } {} { mini-right } {} { mini-right* } {}
2707     }
2708     { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2709 }

```

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

13.29.5 Function for storing content in prop list

```

__enumext_store_addto_prop:n
__enumext_store_addto_prop:V

```

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

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

```

2710 \cs_new_protected:Npn __enumext_store_addto_prop:n #1

```

```

2711 {
2712   \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2713   {
2714     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2715   }
2716   { #1 }
2717 }
2718 \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V }

```

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

13.29.6 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.

```

2719 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2720 {
2721   \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
2722 }
2723 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V }

```

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

13.29.7 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.

```

2724 \cs_new_protected:Npn \__enumext_store_level_open:
2725 {
2726   \bool_if:NT \l__enumext_check_answers_bool
2727   {
2728     \tl_if_empty:CTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2729     {
2730       \__enumext_store_addto_seq:n
2731       {
2732         \item \begin{enumext}
2733       }
2734     }
2735     {
2736       \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2737       {
2738         \item \begin{enumext} [
2739       }
2740       \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2741       {
2742       ]
2743       }
2744       \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2745     }
2746   }
2747 }
2748 \cs_new_protected:Npn \__enumext_store_level_close:
2749 {
2750   \bool_if:NT \l__enumext_check_answers_bool
2751   {
2752     \__enumext_store_addto_seq:n { \end{enumext} }
2753   }
2754 }

```

(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.

```

2755 \cs_new_protected:Npn \__enumext_store_level_open_vii:
2756 {
2757   \bool_if:NT \l__enumext_check_answers_bool
2758   {
2759     \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2760     {

```

```

2761         \__enumext_store_addto_seq:n
2762         {
2763             \item \begin{enumext*}
2764         }
2765     }
2766     {
2767         \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2768         {
2769             \item \begin{enumext*}[
2770         }
2771         \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2772         {
2773             ]
2774         }
2775         \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2776     }
2777 }
2778 }
2779 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2780 {
2781     \bool_if:NT \l__enumext_check_answers_bool
2782     {
2783         \__enumext_store_addto_seq:n { \end{enumext*} }
2784     }
2785 }

```

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

13.29.8 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`

```

2786 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2787 {
2788     \mode_leave_vertical:
2789     \skip_horizontal:n { -\dim_use:N #2 }
2790     \hbox_overlap_left:n
2791     {
2792         \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2793         {
2794             \tl_use:N \l__enumext_mark_answer_sym_tl
2795         }
2796     }
2797     \skip_horizontal:n { \dim_use:N #2 }
2798 }
2799 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }

```

(End of definition for `__enumext_print_keyans_box:NN`.)

13.30 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`.

```

2800 \cs_new_protected:Nn \__enumext_store_internal_ref:
2801 {
2802     \cs_set_protected:Npn \__enumext_tmp:n ##1
2803     {
2804         \tl_set_eq:cc { \l__enumext_label_copy_##1_tl } { \l__enumext_label_##1_tl }
2805         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2806         \tl_remove_once:cn { \l__enumext_label_copy_##1_tl } { . }
2807         \tl_reverse:c { \l__enumext_label_copy_##1_tl }
2808     }
2809     \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2810     \cs_set:Npn \__enumext_tmp:n ##1
2811     { . \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.

```

2812 \bool_lazy_all:nT
2813 {
2814   { \bool_if_p:N \g__enumext_starred_bool }
2815   { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2816 }
2817 {
2818   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2819   { \tl_use:N \l__enumext_label_copy_vii_tl }
2820 }
2821 \bool_lazy_all:nT
2822 {
2823   { \bool_not_p:n { \g__enumext_standar_bool } }
2824   { \bool_if_p:N \l__enumext_standar_bool }
2825   { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2826 }
2827 {
2828   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2829   {
2830     \tl_use:N \l__enumext_label_copy_vii_tl
2831     \int_step_function:nnN { 1 } { \l__enumext_level_int } \l__enumext_tmp:n
2832   }
2833 }

```

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.

```

2834 \bool_lazy_all:nT
2835 {
2836   { \bool_if_p:N \g__enumext_standar_bool }
2837   { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2838   { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2839 }
2840 {
2841   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2842   {
2843     \tl_use:N \l__enumext_label_copy_i_tl
2844     \int_step_function:nnN { 2 } { \l__enumext_level_int } \l__enumext_tmp:n
2845   }
2846 }
2847 \cs_set:Npn \l__enumext_tmp:n ##1
2848 { \tl_use:c { \l__enumext_label_copy_ \int_to_roman:n {##1} _tl } . }
2849 \bool_lazy_all:nT
2850 {
2851   { \bool_if_p:N \g__enumext_standar_bool }
2852   { \bool_if_p:N \l__enumext_starred_bool }
2853   { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2854 }
2855 {
2856   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2857   {
2858     \int_step_function:nnN { 1 } { \l__enumext_level_int } \l__enumext_tmp:n
2859     \tl_use:N \l__enumext_label_copy_vii_tl
2860   }
2861 }

```

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

```

2862 \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2863 {
2864   \l__enumext_store_name_tl \c_colon_str
2865   \int_eval:n { \prop_count:c { \g__enumext_ \l__enumext_store_name_tl _prop } }
2866 }

```

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

```

2867 \tl_put_right:Ne \l__enumext_write_aux_file_tl
2868 {
2869   \l__enumext_newlabel:nn
2870   { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2871   { \l__enumext_newlabel_arg_two_tl }
2872 }
2873 \l__enumext_write_aux_file_tl

```

```
2874 }
```

(End of definition for `__enumext_store_internal_ref:`)

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

```
\__enumext_store_anskey_arg:n
```

The internal function `__enumext_store_anskey_arg: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*” $\langle \textit{argument} \rangle$.

```
2875 \cs_new_protected:Npn \__enumext_store_anskey_arg:n #1
2876 {
2877   \int_gincr:N \g__enumext_item_anskey_int
2878   \__enumext_store_addto_prop:n {#1}
2879   \bool_if:NT \l__enumext_store_ref_key_bool
2880   {
2881     \__enumext_store_internal_ref:
2882   }
2883   \__enumext_anskey_show_wrap_left:n { #1 }
```

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

```
2884 \tl_clear:N \l__enumext_store_anskey_arg_tl
2885 \bool_lazy_and:nnT
2886 { \bool_if_p:N \l__enumext_store_columns_break_bool }
2887 { \bool_not_p:n { \l__enumext_starred_bool } }
2888 {
2889   \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2890 }
2891 \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 $\langle \textit{number} \rangle$ to `\l__enumext_store_anskey_arg_tl`.

```
2892 \bool_lazy_and:nnT
2893 { \bool_not_p:n { \l__enumext_starred_bool } }
2894 { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2895 {
2896   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2897   {
2898     ( \exp_not:V \l__enumext_store_item_join_int )
2899   }
2900 }
```

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

```
2901 \bool_if:NTF \l__enumext_store_item_star_bool
2902 {
2903   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2904   \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2905   {
2906     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2907     {
2908       [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2909     }
2910   }
2911   \dim_compare:nT
2912   {
2913     \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2914   }
2915   {
2916     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2917     {
2918       [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2919     }
2920   }
2921   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2922 }
2923 {
2924   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2925 }
```

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

```

2926 \bool_lazy_and:nnT
2927 { \bool_if_p:N \l__enumext_store_ref_key_bool }
2928 { \bool_if_p:N \l__enumext_hyperref_bool }
2929 {
2930   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2931   {
2932     \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2933     { \exp_not:V \l__enumext_mark_ref_sym_tl }
2934   }
2935 }
2936 \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2937 }

```

(End of definition for `__enumext_store_anskey_arg: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` and `wrap-sep` keys.

```

2938 \cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
2939 {
2940   \par
2941   \bool_if:NTF \l__enumext_starred_bool
2942   {
2943     \dim_compare:nNnT { \l__enumext_mark_sym_sep_dim } = { \c_zero_dim }
2944     {
2945       \dim_set:Nn \l__enumext_mark_sym_sep_dim { \l__enumext_labelsep_vii_dim }
2946     }
2947     \__enumext_print_keyans_box:NN
2948     \l__enumext_labelwidth_vii_dim \l__enumext_mark_sym_sep_dim
2949   }
2950   {
2951     \dim_compare:nNnT { \l__enumext_mark_sym_sep_dim } = { \c_zero_dim }
2952     {
2953       \dim_set:Nn \l__enumext_mark_sym_sep_dim
2954       {
2955         \dim_use:c { \l__enumext_labelsep_ \__enumext_level: _dim }
2956       }
2957     }
2958     \__enumext_print_keyans_box:cc
2959     { \l__enumext_labelwidth_ \__enumext_level: _dim } { \l__enumext_mark_sym_sep_dim }
2960   }
2961   \__enumext_anskey_wrapper:n { #1 }
2962 }

```

(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.

```

2963 \cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
2964 {
2965   \bool_if:NT \l__enumext_show_answer_bool
2966   {
2967     \__enumext_anskey_show_wrap_arg:n { #1 }
2968   }
2969   \bool_if:NT \l__enumext_show_position_bool
2970   {
2971     \tl_set:Ne \l__enumext_mark_answer_sym_tl
2972     {
2973       \group_begin:
2974       \exp_not:N \normalfont
2975       \exp_not:N \footnotesize [ \int_eval:n
2976       {
2977         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2978       }
2979       ]
2980       \group_end:
2981     }

```

```

2982         \__enumext_anskey_show_wrap_arg:n { #1 }
2983     }
2984 }

```

(End of definition for __enumext_anskey_show_wrap_left:n.)

13.32 The command \anskey

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

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

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

```

break-col
item-join
item-star
item-sym*
item-pos*
unknown
\__enumext_anskey_unknown:n
\__enumext_anskey_unknown:nn
2985 \keys_define:nn { enumext / anskey }
2986 {
2987     break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2988     break-col .default:n = true,
2989     break-col .value_forbidden:n = true,
2990     item-join .int_set:N = \l__enumext_store_item_join_int,
2991     item-join .value_required:n = true,
2992     item-star .bool_set:N = \l__enumext_store_item_star_bool,
2993     item-star .default:n = true,
2994     item-star .value_forbidden:n = true,
2995     item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
2996     item-sym* .value_required:n = true,
2997     item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
2998     item-pos* .value_required:n = true,
2999     unknown .code:n = { \__enumext_anskey_unknown:n {#1} },
3000 }

```

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

```

3001 \cs_new_protected:Npn \__enumext_anskey_unknown:n #1
3002 {
3003     \exp_args:NV \__enumext_anskey_unknown:nn \l_keys_key_str {#1}
3004 }
3005 \cs_new_protected:Npn \__enumext_anskey_unknown:nn #1 #2
3006 {
3007     \tl_if_blank:nTF {#2}
3008     {
3009         \msg_error:nnn { enumext } { anskey-cmd-key-unknown } {#1}
3010     }
3011     {
3012         \msg_error:nnnn { enumext } { anskey-cmd-key-value-unknown } {#1} {#2}
3013     }
3014 }

```

(End of definition for `break-col` and others.)

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

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

```

3015 \NewDocumentCommand \anskey { o +m }
3016 {
3017     \__enumext_anskey_safe_outer:
3018     \group_begin:
3019         \bool_if:NT \l__enumext_check_answers_bool
3020         {
3021             \tl_if_novalue:nF {#1}
3022             {
3023                 \keys_set:nn { enumext / anskey } {#1}
3024             }
3025             \tl_if_blank:nTF {#2}
3026             {
3027                 \msg_error:nn { enumext } { anskey-empty-arg }

```

```

3028         }
3029     {
3030         \__enumext_anskey_safe_inner:
3031         \__enumext_store_anskey_arg:n {#2}
3032     }
3033 }
3034 \group_end:
3035 }

```

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

13.32.1 Internal functions for the command

`__enumext_anskey_safe_outer:`
`__enumext_anskey_safe_inner:`

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

```

3036 \cs_new_protected:Nn \__enumext_anskey_safe_outer:
3037 {
3038     \bool_if:NF \l__enumext_store_active_bool
3039     {
3040         \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
3041     }
3042     \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
3043     {
3044         \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
3045     }
3046     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
3047     {
3048         \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans* }
3049     }
3050     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
3051     {
3052         \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
3053     }
3054 }

```

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.

```

3055 \cs_new_protected:Nn \__enumext_anskey_safe_inner:
3056 {
3057     \int_incr:N \l__enumext_anskey_level_int
3058     \int_compare:nNnT { \l__enumext_anskey_level_int } > { 1 }
3059     {
3060         \msg_error:nn { enumext } { anskey-nested }
3061     }
3062     \bool_if:NF \l__enumext_item_number_bool
3063     {
3064         \msg_error:nn { enumext } { anskey-unnumber-item }
3065     }
3066     \mode_if_math:T
3067     {
3068         \msg_error:nne { enumext } { anskey-math-mode } { \c_backslash_str anskey }
3069     }
3070 }

```

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

13.33 The environment `anskey*`

The original implementation of the `anskey*` environment used non-public functions from the `scontents`[4] package, which was not the best approach. Fortunately L^AT_EX release 2025-06-01 implemented the new `c`-type argument in the `\lcmd`[13], with which we can record the *(body)* of the environment in *verbatim mode* and, together with `\scantokens` do the work as the original implementation.

`break-col`
`item-join`
`item-star`
`item-sym*`
`item-pos*`
`force-eol`
`write-env`
`overwrite`
`unknown`

First we add the same keys from the `\anskey` command along with the `force-eol`, `write-env` and `overwrite` keys that were in the original implementation that used the `scontents` support package for these.

```

3071 \keys_define:nn { enumext / anskey* }
3072 {
3073     break-col .bool_set:N = \l__enumext_store_columns_break_bool,
3074     break-col .default:n = true,
3075     break-col .value_forbidden:n = true,
3076     item-join .int_set:N = \l__enumext_store_item_join_int,
3077     item-join .value_required:n = true,

```

```

3078     item-star .bool_set:N = \l__enumext_store_item_star_bool,
3079     item-star .default:n = true,
3080     item-star .value_forbidden:n = true,
3081     item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
3082     item-sym* .value_required:n = true,
3083     item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
3084     item-pos* .value_required:n = true,
3085     force-eol .bool_set:N = \l__enumext_anskey_env_force_eol_bool,
3086     force-eol .initial:n = false,
3087     force-eol .default:n = true,
3088     write-env .code:n = {
3089         \bool_set_true:N \l__enumext_write_anskey_env_bool
3090         \tl_set:Nn \l__enumext_write_anskey_env_file_name_tl {#1}
3091     },
3092     write-env .value_required:n = true,
3093     overwrite .bool_set:N = \l__enumext_anskey_env_overwrite_bool,
3094     overwrite .initial:n = false,
3095     overwrite .default:n = true,
3096     unknown .code:n = { \__enumext_anskey_env_unknown:n {#1} },
3097 }

```

(End of definition for `break-col` and others.)

```

\__enumext_anskey_env_unknown:n
\__enumext_anskey_env_unknown:nn

```

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`.

```

3098 \cs_new_protected:Npn \__enumext_anskey_env_unknown:n #1
3099 {
3100     \exp_args:NV \__enumext_anskey_env_unknown:nn \l_keys_key_str {#1}
3101 }
3102 \cs_new_protected:Npn \__enumext_anskey_env_unknown:nn #1#2
3103 {
3104     \tl_if_blank:nTF {#2}
3105     {
3106         \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
3107     }
3108     {
3109         \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
3110     }
3111 }

```

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

```

\__enumext_anskey_env_file_if_writable:n
\__enumext_anskey_env_file_if_writable:nT
\__enumext_anskey_env_file_if_writable:nF
\__enumext_anskey_env_file_if_writable:nTF

```

The conditional function `__enumext_anskey_env_file_if_writable:n` used by the `write-env` and `overwrite` keys in the `anskey*` environment to determine whether the output file is written or overwritten.

```

3112 \prg_new_protected_conditional:Npnn \__enumext_anskey_env_file_if_writable:n #1 { T, F, TF }
3113 {
3114     \bool_if:NTF \l__enumext_write_anskey_env_bool
3115     {
3116         \file_if_exist:nTF {#1}
3117         {
3118             \bool_if:NTF \l__enumext_anskey_env_overwrite_bool
3119             {
3120                 \msg_warning:nne { enumext } { overwrite-file } {#1}
3121                 \prg_return_true:
3122             }
3123             {
3124                 \msg_warning:nne { enumext } { not-writing } {#1}
3125                 \prg_return_false:
3126             }
3127         }
3128         {
3129             \msg_warning:nne { enumext } { writing-file } {#1}
3130             \prg_return_true:
3131         }
3132     }
3133     { \prg_return_false: }
3134 }

```

The `__enumext_anskey_env_file_write:nn` function is used by the `write-env` key in the `anskey*` environment to write the output file with the *⟨body⟩* of the environment.

```

3135 \cs_new_protected:Npn \__enumext_anskey_env_file_write:nn #1#2

```

```

3136 {
3137   \__enumext_anskey_env_file_if_writable:nT {#1}
3138   {
3139     \iow_open:Nn \__enumext_write_anskey_env_file_iow {#1}
3140     \iow_now:Nn \__enumext_write_anskey_env_file_iow {#2}
3141     \iow_close:N \__enumext_write_anskey_env_file_iow
3142   }
3143 }
3144 \cs_generate_variant:Nn \__enumext_anskey_env_file_write:nn { VV }

```

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

anskey* First, we'll call the function `__enumext_anskey_env_safe_outer:` to make sure where we're running the environment, then, we'll check the state of the variable `__enumext_check_answers_bool` set by the key `no-store`. If it's true, we'll look for `[⟨key = val⟩]` and verify that the *argument* `c` (*body*) is not empty. Finally, we'll run the internal check function `__enumext_anskey_env_safe_inner:n` and call the function `__enumext_store_anskey_arg:n`.

```

3145 \NewDocumentEnvironment{anskey*}{ o c }
3146 {
3147   \__enumext_anskey_env_safe_outer:
3148   \bool_if:NT \__enumext_check_answers_bool
3149   {
3150     \tl_if_novalue:nF {#1}
3151     {
3152       \keys_set:nn { enumext / anskey* } {#1}
3153     }
3154     \tl_if_blank:nTF {#2}
3155     {
3156       \msg_error:nn { enumext } { anskey-empty-arg }
3157     }
3158     {
3159       \__enumext_anskey_env_safe_inner:
3160       \__enumext_store_anskey_env:n {#2}
3161     }
3162   }
3163 } { }

```

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

13.33.1 Internal functions for the environment

`__enumext_anskey_env_safe_outer:` The function `__enumext_store_anskey_safe_outer:` will return the appropriate messages when `anskey*` is executed outside the environment in which the `save-ans` key was activated or within the `keyans`, `keyans*` or `keyanspic` environments.

```

3164 \cs_new_protected:Nn \__enumext_anskey_env_safe_outer:
3165 {
3166   \bool_if:NF \__enumext_store_active_bool
3167   {
3168     \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
3169   }
3170   \int_compare:nNnT { \__enumext_keyans_level_int } = { 1 }
3171   {
3172     \msg_error:nnn { enumext } { anskey-env-wrong } { keyans }
3173   }
3174   \int_compare:nNnT { \__enumext_keyans_level_h_int } = { 1 }
3175   {
3176     \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
3177   }
3178   \int_compare:nNnT { \__enumext_keyans_pic_level_int } = { 1 }
3179   {
3180     \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
3181   }
3182 }

```

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

```

3183 \cs_new_protected:Nn \__enumext_anskey_env_safe_inner:
3184 {
3185   \bool_if:NF \__enumext_item_number_bool
3186   {
3187     \msg_error:nn { enumext } { anskey-unnumber-item }
3188   }

```



```

3189 \mode_if_math:T
3190 {
3191     \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
3192 }
3193 }

```

The `__enumext_store_anskey_env:n` function will first pass the argument `c` (*body*) to the variable `\l__enumext_store_anskey_env_tl` and replace the macro `\obeyedline` with `^^J` and then execute the `write-env` and `overwrite` keys, check the state of the variable `\l__enumext_anskey_env_force_eol_bool` managed by the `force-eol` key and we will add `\c__enumext_anskey_env_hidden_space_str` if necessary. Finally we will use `\exp_args:Ne` on the `__enumext_store_anskey_arg:n` to expand the `__enumext_scan_tokens:n` function which rescans the `\l__enumext_store_anskey_env_tl` variable before processing it.

```

3194 \cs_new_protected:Npn \__enumext_store_anskey_env:n #1
3195 {
3196     \tl_set:Nn \l__enumext_store_anskey_env_tl {#1}
3197     \RenewDocumentCommand \obeyedline { } { \iow_char:N ^^J }
3198     \tl_replace_all:Nee \l__enumext_store_anskey_env_tl { \obeyedline } { \iow_char:N ^^J }
3199     \__enumext_anskey_env_file_write:VV
3200     \l__enumext_write_anskey_env_file_name_tl \l__enumext_store_anskey_env_tl
3201     \bool_if:NF \l__enumext_anskey_env_force_eol_bool
3202     {
3203         \tl_put_right:Nc \l__enumext_store_anskey_env_tl
3204         {
3205             \c__enumext_anskey_env_hidden_space_str
3206         }
3207     }
3208     \exp_args:Ne
3209     \__enumext_store_anskey_arg:n
3210     {
3211         \__enumext_scan_tokens:n { \l__enumext_store_anskey_env_tl }
3212     }
3213 }

```

Since `\obeyedline` can be redefined by the user, for example to `\mbox{}\par`, it is necessary to redefine it to `^^J` in order to use `\tl_replace_all:Nee` otherwise it returns an error.

(End of definition for `__enumext_anskey_env_safe_outer:`, `__enumext_anskey_env_safe_inner:`, and `__enumext_store_anskey_env:n`.)

13.34 Executing check-ans system 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 and finally we execute the function `__enumext_reset_global_vars:` returning the used variables to their original state.

```

3214 \cs_new_protected:Nn \__enumext_execute_after_env:
3215 {
3216     \int_compare:nNtT { \l__enumext_level_int } = { 0 }
3217     {
3218         \tl_if_empty:NF \g__enumext_store_name_tl
3219         {
3220             \__enumext_stop_save_ans_msg:
3221             \__enumext_item_answer_diff:
3222             \__enumext_log_global_vars:
3223             \__enumext_log_answer_vars:
3224             \bool_if:NTF \g__enumext_check_ans_key_bool
3225             {
3226                 \__enumext_check_ans_show:
3227             }
3228             { \__enumext_check_ans_log: }
3229         }
3230         \__enumext_reset_global_vars:
3231     }
3232 }

```

This function is passed to the function `__enumext_after_env:nn` for the environments `enumext` (§13.41) and `enumext*` (§13.46) and it is executed only when the environments are not nested or at some level of these..

(End of definition for `__enumext_execute_after_env:`.)

13.35 Common functions for keyans, keyans* and keyanspic

13.35.1 Storing content in prop list

`__enumext_keyans_addto_prop:n`

The function `__enumext_keyans_addto_prop:n` will pass the the current $\langle label \rangle$ for `\item*` in `keyans` environment and the current $\langle label \rangle$ for `\anspic*` in `keyanspic` environment followed by the $\langle contents \rangle$ 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`.

```

3233 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
3234 {
3235   \tl_clear:N \__enumext_store_current_label_tl
3236   \int_compare:nNnTF { \__enumext_keyans_pic_level_int } = { 1 }
3237   {
3238     \tl_put_right:Ne \__enumext_store_current_label_tl { \__enumext_label_vi_tl }
3239   }
3240   {
3241     \tl_put_right:Ne \__enumext_store_current_label_tl { \__enumext_label_v_tl }
3242   }

```

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

```

3243   \tl_if_novalue:nF { #1 }
3244   {
3245     \tl_if_empty:NF \__enumext_store_keyans_item_opt_sep_v_tl
3246     {
3247       \tl_put_right:NV \__enumext_store_current_label_tl \__enumext_store_keyans_item_opt_sep_v_tl
3248     }
3249     \tl_put_right:Nn \__enumext_store_current_label_tl { #1 }
3250   }
3251   \__enumext_store_addto_prop:V \__enumext_store_current_label_tl
3252 }

```

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

13.35.2 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 $\langle label \rangle$ for `\item*` and `\anspic*` with the $\langle contents \rangle$ of the *optional argument*. The mechanism defined here will allow to execute `\ref{\langle store name : position \rangle}` and will return `1.(A)`.

`__enumext_keyans_store_ref:`

`__enumext_keyans_store_ref_aux_i:`

`__enumext_keyans_store_ref_aux_ii:`

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

```

3253 \cs_new_protected:Nn \__enumext_keyans_store_ref:
3254 {
3255   \bool_if:NT \__enumext_store_ref_key_bool
3256   {
3257     \cs_set_protected:Npn \__enumext_tmp:n ##1
3258     {
3259       \tl_set_eq:cc { \__enumext_label_copy_##1_tl } { \__enumext_label_##1_tl }
3260       \tl_reverse:c { \__enumext_label_copy_##1_tl }
3261       \tl_remove_once:cn { \__enumext_label_copy_##1_tl } { . }
3262       \tl_reverse:c { \__enumext_label_copy_##1_tl }
3263     }
3264     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
3265     \__enumext_keyans_store_ref_aux_i:
3266   }
3267 }

```

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

```

3268 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
3269 {
3270   \bool_if:NT \g__enumext_starred_bool
3271   {
3272     \tl_set_eq:NN \__enumext_label_copy_i_tl \__enumext_label_copy_vii_tl
3273   }
3274   \int_compare:nNnT { \__enumext_keyans_pic_level_int } = { 1 }
3275   {
3276     \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
3277     { \__enumext_label_copy_i_tl . \__enumext_label_copy_vi_tl }

```

```

3278   }
3279   \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
3280   {
3281     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
3282       { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
3283   }
3284   \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
3285   {
3286     \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
3287       { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
3288   }
3289   \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
3290   {
3291     \l__enumext_store_name_tl \c_colon_str
3292     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
3293   }
3294   \__enumext_keyans_store_ref_aux_ii:
3295 }

```

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

```

3296 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
3297 {
3298   \tl_put_right:Ne \l__enumext_write_aux_file_tl
3299   {
3300     \__enumext_newlabel:nn
3301       { \exp_not:V \l__enumext_newlabel_arg_one_tl }
3302       { \l__enumext_newlabel_arg_two_tl }
3303   }
3304   \l__enumext_write_aux_file_tl
3305 }

```

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

13.35.3 Storing content in sequence

```

\__enumext_keyans_addto_seq:n
\__enumext_keyans_addto_seq_link:

```

The function `__enumext_keyans_addto_seq:n` will pass the contents of the current *⟨label⟩* `\l__enumext_label_v_tl` for the *keyans* environment and the `\l__enumext_label_vi_tl` for the *keyanspic* environment when using `\item*` and `\anspic*`, followed by the *⟨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.

```

3306 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
3307 {
3308   \tl_clear:N \l__enumext_store_current_label_tl
3309   \int_compare:nNtF { \l__enumext_keyans_pic_level_int } = { 1 }
3310   {
3311     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
3312   }
3313   {
3314     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
3315   }
3316   \tl_if_novalue:nF { #1 }
3317   {
3318     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_v_tl
3319     {
3320       \tl_put_right:NV \l__enumext_store_current_label_tl \l__enumext_store_keyans_item_opt.
3321     }
3322     \tl_put_right:Nn \l__enumext_store_current_label_tl { #1 }
3323   }
3324   \__enumext_keyans_addto_seq_link:
3325 }

```

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.

```

3326 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
3327 {
3328   \bool_lazy_and:nnT

```

```

3329 { \bool_if_p:N \l__enumext_store_ref_key_bool }
3330 { \bool_if_p:N \l__enumext_hyperref_bool }
3331 {
3332   \tl_put_right:Ne \l__enumext_store_current_label_tl
3333   {
3334     \hfill \exp_not:N \hyperlink
3335     {
3336       \exp_not:V \l__enumext_newlabel_arg_one_tl
3337     }
3338     { \exp_not:V \l__enumext_mark_ref_sym_tl }
3339   }
3340 }
3341 \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
3342 \bool_if:NT \l__enumext_check_answers_bool
3343 {
3344   \int_gincr:N \g__enumext_item_anskey_int
3345 }
3346 }

```

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

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

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`.

```

\__enumext_keyans_save_item_opt:n
\__enumext_keyans_show_item_opt:
\__enumext_keyans_show_item_opt_viii:

```

```

3347 \cs_new_protected:Npn \__enumext_keyans_save_item_opt:n #1
3348 {
3349   \tl_if_novalue:nF { #1 }
3350   {
3351     \tl_set:Nn \l__enumext_store_current_opt_arg_tl { #1 }
3352   }
3353 }

```

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 next to the key `wrap-opt` in `keyans` and `keyanspic` environments.

```

3354 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
3355 {
3356   \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
3357   {
3358     \bool_lazy_or:nnT
3359     { \bool_if_p:N \l__enumext_show_answer_bool }
3360     { \bool_if_p:N \l__enumext_show_position_bool }
3361     {
3362       \__enumext_keyans_wrapper_opt_v:n
3363       { \l__enumext_store_current_opt_arg_tl } \c_space_tl
3364     }
3365   }
3366 }

```

The function `__enumext_keyans_show_item_opt_viii:` will print the optional argument of `\item*` when the `show-ans` or `show-pos` keys are set next to the key `wrap-opt` in `keyans*` environment.

```

3367 \cs_new_protected:Nn \__enumext_keyans_show_item_opt_viii:
3368 {
3369   \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
3370   {
3371     \bool_lazy_or:nnT
3372     { \bool_if_p:N \l__enumext_show_answer_bool }
3373     { \bool_if_p:N \l__enumext_show_position_bool }
3374     {
3375       \__enumext_keyans_wrapper_opt_viii:n
3376       { \l__enumext_store_current_opt_arg_tl } \c_space_tl
3377     }
3378   }
3379 }

```

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

```

\__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`.

```

3380 \cs_new_protected:Nn \__enumext_keyans_pos_mark_set:
3381 {

```

```

3382 \__enumext_label_width_by_box:Nn
3383   \l__enumext_mark_sep_tmpa_dim { \l__enumext_label_v_tl }
3384 \str_case:Vn \l__enumext_align_label_pos_v_str
3385 {
3386   { l }
3387   {
3388     \dim_set:Nn \l__enumext_mark_sep_tmpb_dim { \c_zero_dim }
3389   }
3390   { r }
3391   {
3392     \dim_set:Nn \l__enumext_mark_sep_tmpb_dim
3393       { \l__enumext_labelwidth_v_dim - \l__enumext_mark_sep_tmpa_dim }
3394   }
3395   { c }
3396   {
3397     \dim_set:Nn \l__enumext_mark_sep_tmpb_dim
3398       { 0.5\l__enumext_labelwidth_v_dim - 0.5\l__enumext_mark_sep_tmpa_dim }
3399   }
3400 }

```

Here we set the default values for the key `mark-ans*`, `mark-sep*` and `mark-pos*`.

```

3401 \dim_compare:nNtT { \l__enumext_mark_sym_sep_v_dim } = { \c_zero_dim }
3402 {
3403   \dim_set:Nn \l__enumext_mark_sym_sep_v_dim { \l__enumext_labelsep_v_dim }
3404 }
3405 \tl_set_eq:NN \l__enumext_mark_answer_sym_tl \l__enumext_mark_answer_sym_v_tl
3406 \dim_add:Nn \l__enumext_mark_sym_sep_v_dim { \l__enumext_mark_sep_tmpb_dim }
3407 \str_set_eq:NN \l__enumext_mark_position_str \l__enumext_mark_position_v_str
3408 }

```

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

```

3409 \cs_new_protected:Nn \__enumext_keyans_show_ans:
3410 {
3411   \bool_lazy_all:nT
3412   {
3413     { \bool_if_p:N \l__enumext_show_answer_bool }
3414     { \bool_if_p:N \l__enumext_item_wrap_key_bool }
3415   }
3416   {
3417     \__enumext_keyans_pos_mark_set:
3418     \__enumext_print_keyans_box:NN
3419       \l__enumext_labelwidth_v_dim \l__enumext_mark_sym_sep_v_dim
3420   }
3421 }

```

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.

```

3422 \cs_new_protected:Nn \__enumext_keyans_show_pos:
3423 {
3424   \int_compare:nNtTF { \l__enumext_keyans_level_int } = { 1 }
3425   {
3426     \int_incr:N \l__enumext_show_pos_tmp_int
3427   }
3428   {
3429     \int_zero:N \l__enumext_show_pos_tmp_int
3430   }
3431   \bool_lazy_all:nT
3432   {
3433     { \bool_if_p:N \l__enumext_show_position_bool }
3434     { \bool_if_p:N \l__enumext_item_wrap_key_bool }
3435   }
3436   {
3437     \tl_set:Nn \l__enumext_mark_answer_sym_v_tl
3438     {
3439       \group_begin:
3440       \exp_not:N \normalfont
3441       \exp_not:N \footnotesize [ \int_eval:n
3442         {
3443           \prop_count:c { \g__enumext_ \l__enumext_store_name_tl _prop }
3444           + \l__enumext_show_pos_tmp_int
3445         }

```

```

3446         ]
3447     \group_end:
3448 }
3449 \__enumext_keyans_pos_mark_set:
3450 \__enumext_print_keyans_box:NN
3451     \l__enumext_labelwidth_v_dim \l__enumext_mark_sym_sep_v_dim
3452 }
3453 }

```

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

13.36 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[⟨symbol⟩]` 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`.

```

3454 \cs_new_protected:Npn \__enumext_default_item:n #1
3455 {
3456     \tl_if_novalue:nTF {#1}
3457     {
3458         \bool_if:NT \l__enumext_check_answers_bool
3459         {
3460             \int_gincr:N \g__enumext_item_number_int
3461             \bool_set_true:N \l__enumext_item_number_bool
3462         }
3463         \bool_set_true:c { \l__enumext_wrap_label_ \__enumext_level: _bool }
3464         \__enumext_item_std:w \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
3465     }
3466     {
3467         \bool_set_eq:cc
3468         { \l__enumext_wrap_label_ \__enumext_level: _bool }
3469         { \l__enumext_wrap_label_opt_ \__enumext_level: _bool }
3470         \__enumext_item_std:w {#1} \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
3471     }
3472 }

```

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

`__enumext_item_starred_exec:nn`
`__enumext_item_starred_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`.

```

3473 \cs_new_protected:Npn \__enumext_item_starred_exec:nn #1 #2
3474 {
3475     \tl_if_novalue:nTF {#1}
3476     {
3477         \tl_gset_eq:Nc
3478         \g__enumext_item_symbol_aux_tl { \l__enumext_item_symbol_ \__enumext_level: _tl }
3479     }

```

```

3480     {
3481         \tl_gset:Nn \g__enumext_item_symbol_aux_tl {#1}
3482     }
3483     \tl_if_novalue:nTF {#2}
3484     {
3485         \dim_set_eq:cc
3486         { \__enumext_item_symbol_sep_ \__enumext_level: _dim }
3487         { \__enumext_labelsep_ \__enumext_level: _dim }
3488     }
3489     {
3490         \dim_set:cn { \__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
3491     }
3492     \bool_if:NT \l__enumext_check_answers_bool
3493     {
3494         \int_gincr:N \g__enumext_item_number_int
3495         \bool_set_true:N \l__enumext_item_number_bool
3496     }
3497     \bool_set_true:c { \__enumext_wrap_label_ \__enumext_level: _bool }
3498     \__enumext_item_std:w \tl_use:c { \__enumext_fake_item_indent_ \__enumext_level: _tl }
3499 }

```

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

```

3500 \cs_new_protected:Nn \__enumext_item_starred_exec:
3501 {
3502     \tl_if_empty:cF { \__enumext_item_symbol_ \__enumext_level: _tl }
3503     {
3504         \mode_leave_vertical:
3505         \skip_horizontal:n { -\dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3506         \hbox_overlap_left:n { \g__enumext_item_symbol_aux_tl }
3507         \skip_horizontal:n { \dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3508     }
3509 }

```

(End of definition for `__enumext_item_starred_exec:nn` and `__enumext_item_starred_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 (§13.41).

```

3510 \cs_new_protected:Nn \__enumext_redefine_item:
3511 {
3512     \RenewDocumentCommand \item { s o o }
3513     {
3514         \bool_if:nTF {##1}
3515         {
3516             \__enumext_item_starred_exec:nn {##2} {##3}
3517         }
3518         { \__enumext_default_item:n {##2} }
3519     }
3520 }

```

(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 (§13.41).

```

3521 \cs_new_protected:Nn \__enumext_make_label:
3522 {
3523     \IfDocumentMetadataTF
3524     {
3525         \__enumext_make_label_box:
3526     }
3527     {
3528         \bool_if:NTF \l__enumext_mode_box_bool
3529         {
3530             \__enumext_make_label_box:
3531         }
3532         {
3533             \__enumext_make_label_std:
3534         }
3535     }
3536 }

```


Standard definition when `\DocumentMetadata` is not active.

```

3537 \cs_new_protected:Nn \__enumext_make_label_std:
3538 {
3539   \RenewDocumentCommand \makeLabel { m }
3540   {
3541     \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
3542     \__enumext_item_starred_exec:
3543     \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3544     \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3545     {
3546       \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3547     }
3548     { ##1 }
3549     \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
3550     \tl_gclear:N \g__enumext_item_symbol_aux_tl
3551   }
3552 }

```

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`.

```

3553 \cs_new_protected:Nn \__enumext_make_label_box:
3554 {
3555   \RenewDocumentCommand \makeLabel { m }
3556   {
3557     \strut\smash
3558     {
3559       \makebox
3560       [ \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim } ]
3561       [ \str_use:c { l__enumext_align_label_pos_ \__enumext_level: _str } ]
3562       {
3563         \__enumext_item_starred_exec:
3564         \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3565         \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3566         {
3567           \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3568         }
3569         { ##1 }
3570         \tl_gclear:N \g__enumext_item_symbol_aux_tl
3571       }
3572     } % close smash
3573   }
3574 }

```

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

13.37 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*
3575 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
3576 {
3577   \keys_define:nn { enumext / #1 }
3578   {
3579     item-sym* .tl_set:c = { l__enumext_item_symbol_#2_tl },
3580     item-sym* .value_required:n = true,
3581     item-sym* .initial:n = {\textborn},
3582     item-pos* .dim_set:c = { l__enumext_item_symbol_sep_#2_dim },
3583     item-pos* .value_required:n = true,
3584   }
3585 }
3586 \clist_map_inline:nn
3587 {
3588   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
3589 }
3590 { \__enumext_tmp:nn #1 }

```

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

13.38 Handling unknown keys

At this point in the code I already know that I will NOT add more *⟨keys⟩* for 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).

Well, the paragraph above is not so real, after all I had to add more *⟨keys⟩* than I had planned, not everything turns out the way one thinks in life.

13.38.1 Handling unknown keys for keyans, keyans* and keyanspic

Define and set `unknown` key for `keyans`, `keyans*` and `keyanspic` environments. Here it is necessary to set `\l__enumext_envir_name_tl` in case an `unknown` key is passed using `\setenumext`.

```

unknown
\__enumext_keyans_unknown_keys:n
\__enumext_keyans_unknown_keys:nn
3591 \cs_set_protected:Npn \__enumext_tmp:n #1
3592 {
3593   \keys_define:nn { enumext / #1 }
3594   {
3595     unknown .code:n = {
3596       \tl_set:Nn \l__enumext_envir_name_tl {#1}
3597       \__enumext_keyans_unknown_keys:n {#1}
3598     },
3599   }
3600 }
3601 \clist_map_inline:nn { keyans, keyans*, keyanspic } { \__enumext_tmp:n {#1} }

3602 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:n #1
3603 {
3604   \exp_args:NV \__enumext_keyans_unknown_keys:nn \l_keys_key_str {#1}
3605 }
3606 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:nn #1#2
3607 {
3608   \tl_if_blank:nTF {#2}
3609   {
3610     \msg_error:nne { enumext } { keyans-unknown-key } {#1}
3611   }
3612   {
3613     \msg_error:nnee { enumext } { keyans-unknown-key-value } {#1} {#2}
3614   }
3615 }

```

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

13.38.2 Handling unknown keys for enumext*

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

```

unknown
\__enumext_starred_unknown_keys:n
\__enumext_starred_unknown_keys:nn
3616 \keys_define:nn { enumext / enumext* }
3617 {
3618   unknown .code:n = { \__enumext_starred_unknown_keys:n {#1} },
3619 }

3620 \cs_new_protected:Npn \__enumext_starred_unknown_keys:n #1
3621 {
3622   \exp_args:NV \__enumext_starred_unknown_keys:nn \l_keys_key_str {#1}
3623 }
3624 \cs_new_protected:Npn \__enumext_starred_unknown_keys:nn #1#2
3625 {
3626   \tl_if_blank:nTF {#2}
3627   {
3628     \msg_error:nnn { enumext } { starred-unknown-key } {#1}
3629   }
3630   {
3631     \msg_error:nnnn { enumext } { starred-unknown-key-value } {#1} {#2}
3632   }
3633 }

```

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

13.38.3 Handling unknown keys for enumext

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

```

3634 \cs_set_protected:Npn \__enumext_tmp:n #1
3635 {
3636   \keys_define:nn { enumext / #1 }
3637   {
3638     unknown .code:n = { \__enumext_standar_unknown_keys:n {##1} },
3639   }
3640 }
3641 \clist_map_inline:nn { level-1,level-2,level-3,level-4 } { \__enumext_tmp:n {#1} }

```

Internal functions for handling `unknown` key.

```

3642 \cs_new_protected:Npn \__enumext_standar_unknown_keys:n #1
3643 {
3644   \exp_args:NV \__enumext_standar_unknown_keys:nn \l_keys_key_str {#1}
3645 }
3646 \cs_new_protected:Npn \__enumext_standar_unknown_keys:nn #1#2
3647 {
3648   \tl_if_blank:nTF {#2}
3649   {
3650     \msg_error:nnn { enumext } { standar-unknown-key } {#1}
3651   }
3652   {
3653     \msg_error:nnnn { enumext } { standar-unknown-key-value } {#1} {#2}
3654   }
3655 }

```

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

13.39 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`.

```

3656 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
3657 {
3658   \tl_if_novalue:nTF { #1 }
3659   {
3660     \bool_set_true:N \l__enumext_wrap_label_v_bool
3661     \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
3662   }
3663   {
3664     \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
3665     \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
3666   }
3667 }

```

(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.

```

3668 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
3669 {
3670   \__enumext_keyans_save_item_opt:n { #1 }
3671   \bool_set_true:N \l__enumext_wrap_label_v_bool
3672   \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
3673   \__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.

```

3674   \__enumext_keyans_addto_prop:n { #1 }
3675   \__enumext_keyans_store_ref:
3676   \__enumext_keyans_addto_seq:n { #1 }
3677   \int_gincr:N \g__enumext_check_starred_cmd_int
3678 }

```

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

`\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-ans*` 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 (§13.40).

```

3679 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
3680 {
3681   \RenewDocumentCommand \item { s o }
3682   {
3683     \bool_if:nTF {##1}
3684     {
3685       \bool_set_true:N \l__enumext_item_wrap_key_bool % wrap-ans*
3686       \peek_remove_spaces:n
3687       {
3688         \__enumext_keyans_starred_item:n {##2}
3689       }
3690     }
3691     {
3692       \bool_set_false:N \l__enumext_item_wrap_key_bool
3693       \__enumext_keyans_default_item:n {##2}
3694     }
3695   }
3696 }

```

(End of definition for `\item*` and `__enumext_keyans_redefine_item:`. This function is documented on page 16.)

`__enumext_keyans_make_label:` The function `__enumext_keyans_make_label:` redefine `\make_label` for the keys `mode-box`, `align`, `font`, `wrap-label`, `wrap-label*`, `wrap-ans*` and `\item*` for `keyans` environment. This function are passed to `__enumext_list_arg_two_v:` used in the definition of the `keyans` environment (§13.40).

```

3697 \cs_new_protected:Nn \__enumext_keyans_make_label:
3698 {
3699   \IfDocumentMetadataTF
3700   {
3701     \__enumext_keyans_make_label_box:
3702   }
3703   {
3704     \bool_if:NTF \l__enumext_mode_box_bool
3705     {
3706       \__enumext_keyans_make_label_box:
3707     }
3708     {
3709       \__enumext_keyans_make_label_std:
3710     }
3711   }
3712 }

```

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

```

3713 \cs_new_protected:Npn \__enumext_keyans_wrapper_label:n #1
3714 {
3715   \bool_lazy_all:nT
3716   {
3717     { \bool_if_p:N \l__enumext_wrap_label_v_bool }
3718     { \bool_if_p:N \l__enumext_show_answer_bool }
3719     { \bool_if_p:N \l__enumext_item_wrap_key_bool }
3720     { \cs_if_exist_p:N \__enumext_keyans_wrapper_item_v:n }
3721   }
3722   {
3723     \cs_set_eq:NN \__enumext_wrapper_label_v:n \__enumext_keyans_wrapper_item_v:n
3724   }
3725   \bool_if:NTF \l__enumext_wrap_label_v_bool
3726   {
3727     \__enumext_wrapper_label_v:n { #1 }
3728   }
3729   { #1 }
3730 }

```

Standard definition when `\DocumentMetadata` is not active.

```

3731 \cs_new_protected:Nn \__enumext_keyans_make_label_std:
3732 {
3733   \RenewDocumentCommand \makeLabel { m }
3734   {
3735     \tl_use:N \l__enumext_label_fill_left_v_tl
3736     \__enumext_keyans_show_ans:
3737     \__enumext_keyans_show_pos:
3738     \tl_use:N \l__enumext_label_font_style_v_tl
3739     \__enumext_keyans_wrapper_label:n { ##1 }
3740     \tl_use:N \l__enumext_label_fill_right_v_tl
3741   }
3742 }

```

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

```

3743 \cs_new_protected:Nn \__enumext_keyans_make_label_box:
3744 {
3745   \RenewDocumentCommand \makeLabel { m }
3746   {
3747     \strut\smash
3748     {
3749       \makebox[ \l__enumext_labelwidth_v_dim ][ \l__enumext_align_label_pos_v_str ]
3750       {
3751         \__enumext_keyans_show_ans:
3752         \__enumext_keyans_show_pos:
3753         \tl_use:N \l__enumext_label_font_style_v_tl
3754         \__enumext_keyans_wrapper_label:n { ##1 }
3755       }
3756     }
3757   }
3758 }

```

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

13.40 Second argument of the lists

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

Here we will implement the `__enumext_list_arg_two_X:` function, which will be responsible for setting all the list parameters, the counter, the redefinition of `\item`, `\makeLabel` along with the keys `ref`, `itemindent` and `show-length`.

🔗 In the functions `__enumext_list_arg_two_X:` we will implement the “counter” for the environments, but we do NOT set the “start value” for it to be compatible with *tagged PDF* that should be done later.

13.40.1 Calculation of `\leftmargin` and `\itemindent`

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

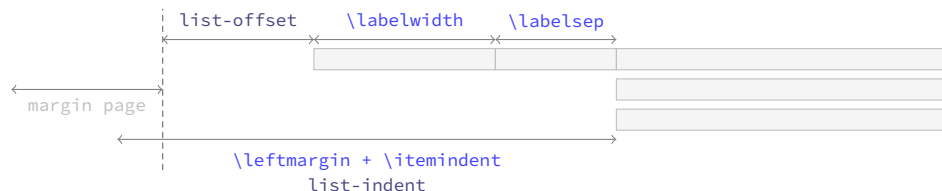


Figure 9: Representation of standard horizontal lengths in `list` environment.

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

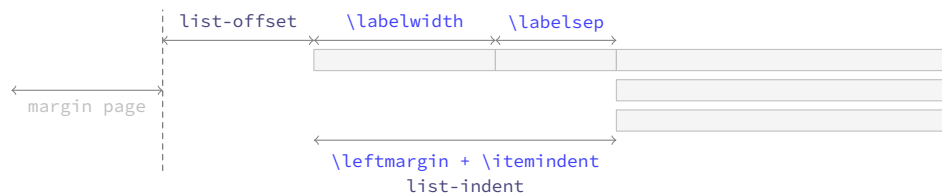


Figure 10: Representation of horizontal lengths concept in list in `enumext`.

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

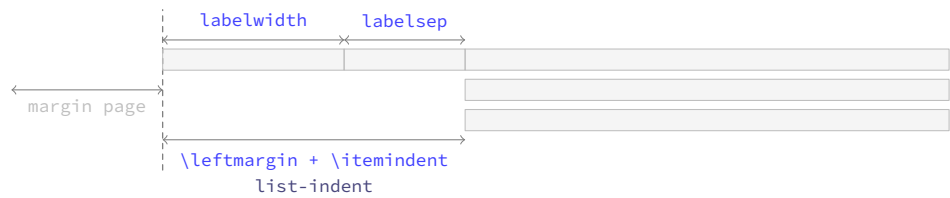


Figure 11: Default horizontal lengths in enumext.

```
\__enumext_calc_hspace:NNNNNNN
\__enumext_calc_hspace:ccccccc
```

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

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

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

```
3759 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
3760 {
3761   \dim_compare:nNt { #1 } < { \c_zero_dim }
3762   {
3763     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
3764     \dim_set:Nn #1 { \dim_abs:n { #1 } }
3765   }
3766   \dim_compare:nNt { #2 } < { \c_zero_dim }
3767   {
3768     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
3769     \dim_set:Nn #2 { \dim_abs:n { #2 } }
3770   }
3771 }
```

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

```
3771   \bool_if:NF #7 { \dim_set:Nn #4 { #1 + #2 } }
```

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

```
3772   \dim_compare:nNtF { #4 } < { \c_zero_dim }
3773   {
3774     \dim_set:Nn #6 { #1 + #2 - #4 }
3775     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3776   }
3777   {
3778     \dim_compare:nNt { #4 } = { #1 + #2 }
3779     { \dim_set:Nn #6 { \c_zero_dim } }
3780     \dim_compare:nNt { #4 } < { #1 + #2 }
3781     { \dim_set:Nn #6 { #1 + #2 - #4 } }
3782     \dim_compare:nNt { #4 } > { #1 + #2 }
3783     {
3784       \dim_set:Nn #6 { -#1 - #2 + #4 }
3785       \dim_set:Nn #6 { #6*-1 }
3786     }
3787     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3788   }
3789 }
3790 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { ccccccc }
```

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

13.40.2 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.

```
3791 \cs_set_protected:Npn \__enumext_tmp:n #1
3792 {
3793   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3794   {
3795     \__enumext_calc_hspace:ccccccc
3796     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3797     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3798     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3799     { \__enumext_leftmargin_tmp_#1_bool }
```

```

3800 \clist_map_inline:nn
3801 { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3802 { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
3803 \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3804 { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
3805 \clist_map_inline:nn { beginparpenalty, itempenalty, endparpenalty }
3806 { \int_set_eq:cc {@####1} { l__enumext_####1_#1_int } }
3807 \usecounter { enumX#1 }
3808 \str_if_eq:nnTF {#1} { v }
3809 {
3810   \__enumext_keyans_redefine_item:
3811   \__enumext_keyans_make_label:
3812   \__enumext_keyans_ref:
3813   \__enumext_keyans_fake_item_indent:
3814   \bool_if:cT { l__enumext_show_length_#1_bool }
3815   {
3816     \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3817   }
3818 }
3819 {
3820   \__enumext_redefine_item:
3821   \__enumext_make_label:
3822   \__enumext_standar_ref:
3823   \__enumext_fake_item_indent:
3824   \bool_if:cT { l__enumext_show_length_#1_bool }
3825   {
3826     \msg_term:nnne { enumext } { list-lengths } {#1}
3827     { \int_use:N \l__enumext_level_int }
3828   }
3829 }
3830 }
3831 }
3832 \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:
\__enumext_list_arg_two_viii:

```

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.

```

3833 \cs_set_protected:Npn \__enumext_tmp:n #1
3834 {
3835   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3836   {
3837     \bool_set_true:c { l__enumext_leftmargin_tmp_#1_bool }
3838     \dim_zero:c { l__enumext_leftmargin_tmp_#1_dim }
3839     \__enumext_calc_hspace:ccccc
3840     { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3841     { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3842     { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3843     { l__enumext_leftmargin_tmp_#1_bool }
3844     \clist_map_inline:nn
3845     { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3846     { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
3847     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3848     { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
3849     \clist_map_inline:nn { beginparpenalty, itempenalty, endparpenalty }
3850     { \int_set_eq:cc {@####1} { l__enumext_####1_#1_int } }
3851     \skip_set_eq:Nc \parsep { l__enumext_itemsep_#1_skip }
3852     \skip_zero:N \partopsep
3853     \usecounter { enumX#1 }
3854     \__enumext_starred_ref:
3855     \str_if_eq:nnTF {#1} { vii }
3856     {
3857       \__enumext_fake_item_indent_vii:
3858       \bool_if:cT { l__enumext_show_length_vii_bool }
3859       { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3860     }
3861     {
3862       \__enumext_fake_item_indent_viii:
3863       \bool_if:cT { l__enumext_show_length_#1_bool }

```



```

3864         { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3865     }
3866 }
3867 }
3868 \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:.)

13.41 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”.

```

3869 \cs_new_protected:Nn \__enumext_safe_exec:
3870 {
3871     \__enumext_is_not_nested:
3872     \__enumext_internal_mini_page:
3873     \int_incr:N \l__enumext_level_int
3874     \int_compare:nNnT { \l__enumext_level_int } > { 4 }
3875     { \msg_fatal:nn { enumext } { list-too-deep } }
3876     \bool_set_true:N \l__enumext_standar_bool
3877     \bool_set_false:N \l__enumext_starred_bool
3878     \__enumext_is_on_first_level:
3879 }

```

(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_name_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.

```

3880 \cs_new_protected:Npn \__enumext_parse_keys:n #1
3881 {
3882     \tl_if_novalue:nF {#1}
3883     {
3884         \str_clear:N \l__enumext_series_name_str
3885         \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
3886         {
3887             \keys_set:nn { enumext / level-1 } {#1}
3888             \bool_if:NF \l__enumext_print_keyans_cmd_bool
3889             {
3890                 \__enumext_parse_series:n {#1}
3891             }
3892             \__enumext_nested_base_line_fix:
3893         }
3894         {
3895             \exp_args:Ne \keys_set:nn
3896             { enumext / level-\int_use:N \l__enumext_level_int } {#1}
3897             \bool_if:NF \l__enumext_print_keyans_cmd_bool
3898             {
3899                 \__enumext_parse_series:n {#1}
3900             }
3901         }
3902         \__enumext_store_active_keys:n {#1}
3903     }
3904 }

```

(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*`.

```

3905 \cs_new_protected:Nn \__enumext_start_store_level:
3906 {
3907     \bool_lazy_all:nT
3908     {
3909         \bool_if_p:N \l__enumext_store_active_bool }

```

```

3910     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3911     { \bool_if_p:N \g__enumext_standar_bool }
3912   }
3913   {
3914     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3915     {
3916       \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3917       \__enumext_store_level_open:
3918     }
3919   }

```

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

```

3920   \bool_lazy_all:nT
3921   {
3922     { \bool_if_p:N \l__enumext_store_active_bool }
3923     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3924     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
3925   }
3926   {
3927     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3928     {
3929       \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3930       \__enumext_store_level_open:
3931     }
3932   }
3933 }

```

(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*`.

```

3934 \cs_new_protected:Nn \__enumext_stop_store_level:
3935 {
3936   \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
3937   {
3938     \__enumext_store_level_close:
3939   }
3940 }

```

(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.

```

3941 \cs_new_protected:Nn \__enumext_multicols_start:
3942 {
3943   \int_compare:nNnT
3944   { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3945   {
3946     \dim_compare:nNnT
3947     { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3948     {
3949       \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
3950       {
3951         ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
3952         + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
3953         ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
3954         - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3955       }
3956     }
3957     \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
3958     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3959     {
3960       \dim_zero:N \columnseprule
3961     }

```

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.

```

3962   \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3963   {
3964     \skip_zero:N \multicolsep
3965     \__enumext_multi_addvspace:

```

```

3966     }
3967     \raggedcolumns
3968     \begin{multicols}{\int_use:c { \l__enumext_columns_ \l__enumext_level: _int } }
3969   }
3970 }

```

(End of definition for \l__enumext_multicols_start:.)

\l__enumext_multicols_stop: The function \l__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 \l__enumext_stop_store_level:.

```

3971 \cs_new_protected:Nn \l__enumext_multicols_stop:
3972 {
3973   \int_compare:nNnTF
3974   { \int_use:c { \l__enumext_columns_ \l__enumext_level: _int } } > { 1 }
3975   {
3976     \l__enumext_stop_list:
3977     \l__enumext_stop_store_level:
3978     \end{multicols}
3979     \l__enumext_unskip_unkern:
3980     \l__enumext_unskip_unkern:
3981     \par\addvspace{ \skip_use:c { \l__enumext_multicols_below_ \l__enumext_level: _skip } }
3982   }
3983   {
3984     \l__enumext_stop_list:
3985     \l__enumext_stop_store_level:
3986   }
3987 }

```

(End of definition for \l__enumext_multicols_stop:.)

\l__enumext_before_list: The function \l__enumext_before_list: first calls the function \l__enumext_vspace_above: used by the keys `above` and `above*`, then calls the function \l__enumext_before_args_exec: used by the key `before*` and finally execute the function \l__enumext_check_ans_active: for the check answer mechanism.

```

3988 \cs_new_protected:Nn \l__enumext_before_list:
3989 {
3990   \l__enumext_vspace_above:
3991   \l__enumext_before_args_exec:
3992   \l__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.

```

3993   \dim_compare:nNnT
3994   { \dim_use:c { \l__enumext_minipage_right_ \l__enumext_level: _dim } } > { \c_zero_dim }
3995   {
3996     \dim_set:cn { \l__enumext_minipage_left_ \l__enumext_level: _dim }
3997     {
3998       \linewidth
3999       - \dim_use:c { \l__enumext_minipage_right_ \l__enumext_level: _dim }
4000       - \dim_use:c { \l__enumext_minipage_hsep_ \l__enumext_level: _dim }
4001     }

```

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 \l__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 \l__enumext_multicols_start: is called to handle the `multicols` environment.

```

4002     \bool_set_true:c { \l__enumext_minipage_active_ \l__enumext_level: _bool }
4003     \int_gincr:N \g__enumext_minipage_stat_int
4004     \l__enumext_minipage_add_space:
4005     \noindent
4006     \l__enumext_mini_page{ \dim_use:c { \l__enumext_minipage_left_ \l__enumext_level: _dim } }
4007   }
4008   \l__enumext_multicols_start:
4009 }

```

(End of definition for \l__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.

```

4010 \cs_new_protected:Nn \__enumext_second_part:
4011 {
4012   \bool_if:cTF { \l__enumext_minipage_active_ \__enumext_level: _bool }
4013   {
4014     \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
4015     {
4016       \msg_warning:nn { enumext } { missing-miniright }
4017       \miniright
4018     }
4019     \int_gzero:N \g__enumext_minipage_stat_int
4020     \__enumext_unskip_unkern: % remove topsep + [partopsep]
4021     \end__enumext_mini_page
4022   }
4023   {
4024     \__enumext_multicols_stop:
4025   }

```

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.

```

4026   \__enumext_after_stop_list:
4027   \__enumext_check_ans_key_hook:
4028   \__enumext_vspace_below:
4029   \bool_set_false:N \l__enumext_standar_bool
4030   \bool_if:NF \l__enumext_print_keyans_cmd_bool
4031   {
4032     \__enumext_standar_save_counter:
4033   }
4034 }

```

(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.

```

4035 \cs_new_protected:Nn \__enumext_set_item_width:
4036 {
4037   \dim_set:Nn \itemwidth { \linewidth }
4038   \dim_compare:nT
4039   {
4040     \dim_use:c { \__enumext_listoffset_ \__enumext_level: _dim } != \c_zero_dim
4041   }
4042   {
4043     \dim_sub:Nn \itemwidth
4044     {
4045       \dim_use:c { \__enumext_listoffset_ \__enumext_level: _dim }
4046     }
4047   }
4048 }

```

(End of definition for `__enumext_set_item_width:`.)

`__enumext_start_counter:` For compatibility with *tagged* PDF and since we are using legacy code for the implementation, we must set the initial value of the counters after the second argument to the list environment and before the first execution of `\item`, i.e. `\begin{list}{\langle arg one \rangle}{\langle arg two \rangle}\setcounter{enumX}`.

🔗 This is described in [processing order of legacysetupcode in the block templates](#) and we will apply the workaround provided by Frank Mittelbach.

```

4049 \cs_new_protected:Nn \__enumext_start_counter:
4050 {
4051   \setcounter { enumX \__enumext_level: }
4052   {
4053     \int_eval:n { \int_use:c { \__enumext_start_ \__enumext_level: _int } - 1 }
4054   }
4055 }

```

(End of definition for `__enumext_start_counter:`.)

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

```

4056 \NewDocumentEnvironment{enumext}{0}{ }
4057 {
4058   \__enumext_safe_exec:
4059   \__enumext_parse_keys:n {#1}
4060   \__enumext_before_list:
4061   \__enumext_start_store_level:
4062   \__enumext_start_list:nn
4063   { \tl_use:c { \__enumext_label_ \__enumext_level: _tl } }
4064   {
4065     \use:c { __enumext_list_arg_two_ \__enumext_level: : }
4066     \__enumext_before_keys_exec:
4067   }
4068   \__enumext_start_counter:
4069   \__enumext_set_item_width:
4070   \__enumext_after_args_exec:
4071 }
4072 {
4073   \__enumext_second_part:
4074 }

```

(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`.

```

4075 \__enumext_after_env:nn {enumext}
4076 {
4077   \__enumext_execute_after_env:
4078 }

```

13.42 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.

```

4079 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
4080 {
4081   \bool_if:NF \__enumext_store_active_bool
4082   {
4083     \msg_error:nnnn { enumext } { wrong-place } { keyans } { save-ans }
4084   }
4085   \int_incr:N \__enumext_keyans_level_int
4086   \bool_set_true:N \__enumext_keyans_env_bool
4087   \__enumext_keyans_name_and_start:
4088   % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
4089   \bool_set_false:N \__enumext_store_active_bool
4090   \int_compare:nNnT { \__enumext_keyans_level_int } > { 1 }
4091   {
4092     \msg_error:nn { enumext } { keyans-nested }
4093   }
4094   \int_compare:nNnT { \__enumext_level_int } > { 1 }
4095   {
4096     \msg_error:nn { enumext } { keyans-wrong-level }
4097   }
4098 }

```

(End of definition for `__enumext_keyans_safe_exec:`.)

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

```

4099 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
4100 {
4101   \keys_set:nn { enumext / keyans } {#1}
4102 }

```

(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:
4103 \cs_new_protected:Nn \__enumext_before_list_v:
4104 {
4105   \__enumext_vspace_above_v:
4106   \__enumext_before_args_exec_v:
4107   \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
4108   {
4109     \dim_set:Nn \l__enumext_minipage_left_v_dim
4110     {
4111       \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
4112     }
4113     \bool_set_true:N \l__enumext_minipage_active_v_bool
4114     \int_gincr:N \g__enumext_minipage_stat_int
4115     \__enumext_keyans_minipage_add_space:
4116     \__enumext_mini_page{ \l__enumext_minipage_left_v_dim }
4117   }
4118   \__enumext_keyans_multicols_start:
4119 }
4120 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
4121 {
4122   \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
4123   {
4124     \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
4125     {
4126       \dim_set:Nn \l__enumext_columns_sep_v_dim
4127       {
4128         (
4129           \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
4130         ) / \l__enumext_columns_v_int
4131         - \l__enumext_listoffset_v_dim
4132       }
4133     }
4134     \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim
4135     \dim_zero:N \columnseprule % no rule here
4136     \bool_if:NF \l__enumext_minipage_active_v_bool
4137     {
4138       \skip_zero:N \multicolsep
4139       \__enumext_keyans_multi_addvspace:
4140     }
4141     \raggedcolumns
4142     \begin{multicols}{\l__enumext_columns_v_int}
4143   }
4144 }
4145 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
4146 {
4147   \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
4148   {
4149     \__enumext_stop_list:
4150     \end{multicols}
4151     \__enumext_unskip_unkern:
4152     \__enumext_unskip_unkern:
4153     \par\addvspace{ \l__enumext_multicols_below_v_skip }
4154   }
4155   {
4156     \__enumext_stop_list:
4157   }
4158 }
4159 \cs_new_protected:Nn \__enumext_second_part_v:
4160 {
4161   \bool_if:NTF \l__enumext_minipage_active_v_bool
4162   {
4163     \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
4164     {
4165       \msg_warning:nn { enumext } { missing-miniright }
4166       \miniright
4167     }
4168     \int_gzero:N \g__enumext_minipage_stat_int
4169     \__enumext_unskip_unkern: % remove \topsep + [\partopsep]
4170     \end__enumext_mini_page
4171     \par\addvspace{ \l__enumext_minipage_after_skip }
4172   }

```

```

4173     {
4174         \__enumext_keyans_multicols_stop:
4175     }
4176     \bool_set_false:N \__enumext_keyans_env_bool
4177     \__enumext_after_stop_list_v:
4178     \__enumext_vspace_below_v:
4179 }

```

(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.

```

4180 \cs_new_protected:Nn \__enumext_keyans_set_item_width:
4181 {
4182     \dim_set:Nn \itemwidth { \linewidth }
4183     \dim_compare:nT
4184     {
4185         \__enumext_listoffset_v_dim != \c_zero_dim
4186     }
4187     {
4188         \dim_sub:Nn \itemwidth { \__enumext_listoffset_v_dim }
4189     }
4190 }

```

(End of definition for __enumext_keyans_set_item_width:.)

__enumext_keyans_start_counter:

For compatibility with *tagged* PDF and since we are using legacy code for the implementation, we must set the initial value of the counters after the second argument to the list environment and before the first execution of \item, i.e. \begin{list}{\langle arg one \rangle}{\langle arg two \rangle}\setcounter{enumX}.

```

4191 \cs_new_protected:Nn \__enumext_keyans_start_counter:
4192 {
4193     \setcounter { enumXv } { \int_eval:n { \int_use:c { l__enumext_start_v_int } - 1 } }
4194 }

```

(End of definition for __enumext_keyans_start_counter:.)

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

```

4195 \NewDocumentEnvironment{keyans}{0}{ }
4196 {
4197     \__enumext_keyans_safe_exec:
4198     \__enumext_keyans_parse_keys:n {#1}
4199     \__enumext_before_list_v:
4200     \__enumext_start_list:nn
4201     { \tl_use:N \__enumext_label_v_tl }
4202     {
4203         \__enumext_list_arg_two_v:
4204         \__enumext_before_keys_exec_v:
4205     }
4206     \__enumext_keyans_start_counter:
4207     \__enumext_keyans_set_item_width:
4208     \__enumext_after_args_exec_v:
4209 }
4210 {
4211     \__enumext_check_starred_cmd:n { item }
4212     \__enumext_second_part_v:
4213 }

```

(End of definition for keyans. This function is documented on page 16.)

13.43 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`[18] and `ltsockets`[20]. 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](#).

13.43.1 Socket for tagging support in 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
```

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

```
4214 \socket_new:nn {tagsupport/__enumext/starred}{ 1 }
4215 \socket_new_plug:nnn {tagsupport/__enumext/starred} {start-list-tags}
4216 {
4217   \tag_resume:n {#1}
4218   \tag_mc_end_push:
4219     \tag_struct_begin:n {tag=LI}
4220     \tag_struct_begin:n {tag=Lbl}
4221     \tag_mc_begin:n {tag=Lbl}
4222 }
4223 \socket_new_plug:nnn {tagsupport/__enumext/starred} {stop-start-tags}
4224 {
4225   \tag_mc_end:
4226   \tag_struct_end:n {tag=Lbl}
4227   \tag_struct_begin:n {tag=LBody}
4228   \tag_struct_begin:n {tag=text-unit}
4229   \tag_struct_begin:n {tag=text}
4230 }
4231 \socket_new_plug:nnn {tagsupport/__enumext/starred} {stop-list-tags}
4232 {
4233   \tag_struct_end:n {tag=text}
4234   \tag_struct_end:n {tag=text-unit}
4235   \tag_struct_end:n {tag=LBody}
4236   \tag_struct_end:n {tag=LI}
4237   \tag_mc_begin_pop:n {}
4238   \tag_suspend:n {#1}
4239 }
```

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

```
4240 \cs_new_protected_nopar:Npn __enumext_start_list_tag:n #1
4241 {
4242   \IfDocumentMetadataT
4243   {
4244     \socket_assign_plug:nn {tagsupport/__enumext/starred} {start-list-tags}
4245     \socket_use:nn {tagsupport/__enumext/starred} {#1}
4246   }
4247 }
4248 \cs_new_protected_nopar:Npn __enumext_stop_start_list_tag:
4249 {
4250   \IfDocumentMetadataT
4251   {
4252     \socket_assign_plug:nn {tagsupport/__enumext/starred} {stop-start-tags}
4253     \socket_use:nn {tagsupport/__enumext/starred} {}
4254   }
4255 }
4256 \cs_new_protected_nopar:Npn __enumext_stop_list_tag:n #1
4257 {
4258   \IfDocumentMetadataT
4259   {
4260     \socket_assign_plug:nn {tagsupport/__enumext/starred} {stop-list-tags}
4261     \socket_use:nn {tagsupport/__enumext/starred} {#1}
4262   }
4263 }
```

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

13.43.2 Socket for tagging support in keyanspic

```
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:
```

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

```
4264 \socket_new:nn {tagsupport/__enumext/keyanspic}{ 0 }
4265 \socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {start-list-tags}
4266 {
4267   \tag_resume:n {keyanspic}
4268   \tag_mc_end_push:
4269     \tag_struct_begin:n {tag=LI}
4270     \tag_struct_begin:n {tag=Lbl}
4271     \tag_mc_begin:n {tag=Lbl}
4272 }
4273 \socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {stop-start-tags}
4274 {
4275   \tag_mc_end:
```

```
4276 \tag_struct_end:n {tag=Lbl}  
4277 \tag_struct_begin:n {tag=LBody}  
4278 \tag_struct_begin:n {tag=text-unit}  
4279 \tag_struct_begin:n {tag=text}  
4280 \tag_mc_begin:n {tag=text}  
4281 }  
4282 \socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {stop-list-tags}  
4283 {  
4284 \tag_mc_end:  
4285 \tag_struct_end:n {tag=text}  
4286 \tag_struct_end:n {tag=text-unit}  
4287 \tag_struct_end:n {tag=LBody}  
4288 \tag_struct_end:n {tag=LI}  
4289 \tag_mc_begin_pop:n {}  
4290 \tag_suspend:n {keyanspic}  
4291 }
```

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

```
4292 \cs_new_protected_nopar:Nn \__enumext_anspic_start_list_tag:  
4293 {  
4294 \IfDocumentMetadataT  
4295 {  
4296 \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {start-list-tags}  
4297 \socket_use:n {tagsupport/__enumext/keyanspic}  
4298 }  
4299 }  
4300 \cs_new_protected_nopar:Nn \__enumext_anspic_stop_start_list_tag:  
4301 {  
4302 \IfDocumentMetadataT  
4303 {  
4304 \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {stop-start-tags}  
4305 \socket_use:n {tagsupport/__enumext/keyanspic}  
4306 }  
4307 }  
4308 \cs_new_protected_nopar:Nn \__enumext_anspic_stop_list_tag:  
4309 {  
4310 \IfDocumentMetadataT  
4311 {  
4312 \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {stop-list-tags}  
4313 \socket_use:n {tagsupport/__enumext/keyanspic}  
4314 }  
4315 }
```

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

13.44 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.

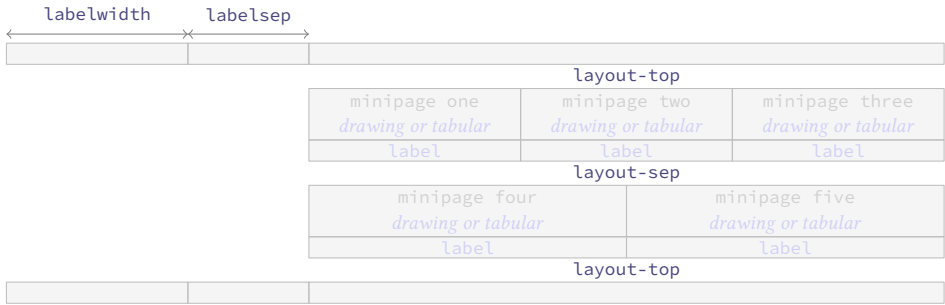


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

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?](#).

13.44.1 The environment keyanspic

label-pos
label-sep
layout-sty
layout-con

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`.

```

4316 \keys_define:nn { enumext / keyanspic }
4317 {
4318   label-pos .choice:,
4319   label-pos / above .code:n =
4320     \bool_set_true:N \l__enumext_anspic_label_above_bool
4321     \str_set:Nn \l__enumext_anspic_mini_pos_str { t },
4322   label-pos / below .code:n =
4323     \bool_set_false:N \l__enumext_anspic_label_above_bool
4324     \str_set:Nn \l__enumext_anspic_mini_pos_str { b },
4325   label-pos / unknown .code:n =
4326     \msg_error:nneee { enumext } { unknown-choice }
4327     { label-pos } { above,~ below } { \exp_not:n {#1} },
4328   label-pos .initial:n = below,
4329   label-pos .value_required:n = true,
4330   label-sep .skip_set:N = \l__enumext_anspic_label_sep_skip,
4331   label-sep .value_required:n = true,
4332   layout-sty .tl_set:N = \l__enumext_anspic_layout_style_tl,
4333   layout-sty .value_required:n = true,
4334   layout-sep .code:n = \keys_set:nn { enumext / keyans } { parsep = #1 },
4335   layout-sep .value_required:n = true,
4336   layout-top .code:n = \keys_set:nn { enumext / keyans } { topsep = #1 },
4337   layout-top .value_required:n = true,
4338   mark-ans .code:n = \keys_set:nn { enumext / keyans } { mark-ans = #1 },
4339   mark-ans .value_required:n = true,
4340   mark-pos .code:n = \keys_set:nn { enumext / keyans } { mark-pos = #1 },
4341   mark-pos .value_required:n = true,
4342   mark-sep .code:n = \keys_set:nn { enumext / keyans } { mark-sep = #1 },
4343   mark-sep .value_required:n = true,
4344   save-sep .code:n = \keys_set:nn { enumext / keyans } { save-sep = #1 },
4345   save-sep .value_required:n = true,
4346   wrap-opt .code:n = \keys_set:nn { enumext / keyans } { wrap-opt = #1 },
4347   wrap-opt .value_required:n = true,
4348   wrap-ans* .code:n = \keys_set:nn { enumext / keyans } { wrap-ans* = #1 },
4349   wrap-ans* .value_required:n = true,
4350   show-ans .code:n = \keys_set:nn { enumext / keyans } { show-ans = #1 },
4351   show-ans .value_required:n = true,
4352   show-pos .code:n = \keys_set:nn { enumext / keyans } { show-pos = #1 },
4353   show-pos .value_required:n = true,
4354   unknown .code:n = {
4355     \tl_set:Nn \l__enumext_envir_name_tl { keyanspic }
4356     \__enumext_keyans_unknown_keys:n {#1}
4357   },
4358 }

```

(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.

```

4359 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
4360 {
4361   \int_incr:N \l__enumext_keyans_pic_level_int
4362   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
4363   {
4364     \msg_error:nn { enumext } { keyanspic-nested }
4365   }
4366   \__enumext_keyans_name_and_start:
4367 }

```

Parse `[(key = val)]` for `keyanspic` environment.

```

4368 \cs_new_protected:Npn \__enumext_keyans_pic_parse_keys:n #1
4369 {
4370   \tl_if_novalue:nF {#1}
4371   {
4372     \keys_set:nn { enumext / keyanspic } {#1}
4373   }
4374 }

```

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

```

4375 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
4376 {
4377   \dim_compare:nNtT { #1 } < { \c_zero_dim }
4378   {
4379     \skip_set:Nn #1 { -#1 }
4380   }
4381 }

```

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.

```

4382 \cs_new_protected:Npn \__enumext_keyans_pic_arg_two:
4383 {
4384   \bool_set_false:N \__enumext_leftmargin_tmp_v_bool
4385   \__enumext_list_arg_two_v:
4386   \__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*.

```

4387   \bool_if:NF \__enumext_anspic_label_above_bool
4388   {
4389     \stepcounter { enumXv }
4390     \hbox_set:Nn \__enumext_anspic_label_box { \__enumext_label_v_tl }
4391     \dim_set:Nn \__enumext_anspic_label_htdp_dim
4392     {
4393       \box_ht_plus_dp:N \__enumext_anspic_label_box
4394     }
4395     \skip_add:Nn \parsep
4396     {
4397       \__enumext_anspic_label_htdp_dim
4398       + \box_dp:N \strutbox
4399       + \__enumext_anspic_label_sep_skip
4400     }
4401   }

```

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.

```

4402   \dim_add:Nn \leftmargin { -\__enumext_labelwidth_v_dim - \__enumext_labelsep_v_dim }
4403   \ignorespaces
4404   \skip_add:Nn \topsep { 0.5\box_dp:N \strutbox }
4405   \dim_zero:N \listparindent
4406   \skip_zero:N \partopsep
4407   \skip_zero:N \itemsep
4408 }

```

(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.

```

4409 \NewDocumentEnvironment{keyanspic}{ o }
4410 {
4411   \__enumext_keyans_pic_safe_exec:
4412   \__enumext_keyans_pic_parse_keys:n {#1}
4413   \begin{list} { } { \__enumext_keyans_pic_arg_two: }
4414   \IfDocumentMetadataT
4415   {
4416     \tag_suspend:n {list}
4417   }
4418   \item[] \scan_stop:
4419   \RenewDocumentCommand \item {}
4420   {
4421     \msg_error:nn { enumext } { keyanspic-item-cmd }
4422   }

```

```

4423 \IfDocumentMetadataT
4424 {
4425   \tag_resume:n {keyanspic}
4426   \tag_tool:n {para/tagging=false}
4427   \tag_suspend:n {keyanspic}
4428 }
4429 }

```

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.

```

4430 {
4431   \IfDocumentMetadataT
4432   {
4433     \tag_resume:n {keyanspic}
4434     \tag_mc_end_push:
4435     \tag_struct_begin:n {tag=L,attribute=enumerate}
4436   }
4437   \__enumext_anspic_exec:
4438   \IfDocumentMetadataT
4439   {
4440     \tag_suspend:n {keyanspic}
4441   }
4442   \end{list}
4443   \IfDocumentMetadataT
4444   {
4445     \tag_struct_end:n {tag=L}
4446     \tag_mc_begin_pop:n {}
4447     \tag_struct_end:n {tag=L}
4448     \tag_mc_begin_pop:n {}
4449   }

```

Finally we check if `\anspic*` has been used, set the counter `enumXvi` to zero and apply our “adjusted” vertical space bottom.

```

4450 \__enumext_check_starred_cmd:n { anspic }
4451 \setcounter { enumXvi } { 0 }
4452 \bool_if:NTF \__enumext_anspic_label_above_bool
4453 {
4454   \par\addvspace{ 0.5\box_dp:N \strutbox }
4455 }
4456 {
4457   \par
4458   \addvspace
4459   {
4460     \dim_eval:n
4461     {
4462       \__enumext_anspic_label_htdp_dim + \box_ht_plus_dp:N \strutbox
4463       + \__enumext_anspic_label_sep_skip + \__enumext_topsep_v_skip
4464     }
4465   }
4466 }
4467 }

```

(End of definition for `keyanspic`. This function is documented on page 17.)

13.44.2 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 `\l__enumext_anspic_args_seq` which is processed by the `keyanspic` environment.

```

4468 \NewDocumentCommand \anspic { s o +m }
4469 {
4470   \bool_if:NF \l__enumext_store_active_bool
4471   {

```

```

4472     \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
4473   }
4474   \int_compare:nNt { \l__enumext_level_int } > { 1 }
4475   {
4476     \msg_error:nn { enumext } { keyanspic-wrong-level }
4477   }
4478   \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
4479   {
4480     \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
4481   }
4482   \seq_put_right:Nn \l__enumext_anspic_args_seq
4483   {
4484     \__enumext_anspic_args:nnn { #1 } { #2 } { #3 }
4485   }
4486 }

```

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”.

```

4487 \cs_new_protected:Npn \__enumext_anspic_body_dim:n #1
4488 {
4489   \bool_if:NF \l__enumext_anspic_label_above_bool
4490   {
4491     \IfDocumentMetadataT
4492     {
4493       \tag_suspend:n {keyanspic}
4494     }
4495     \vbox_set:Nn \l__enumext_anspic_body_box { #1 }
4496     \dim_set:Nn \l__enumext_anspic_body_htdp_dim
4497     {
4498       \box_ht_plus_dp:N \l__enumext_anspic_body_box
4499     }
4500     \IfDocumentMetadataT
4501     {
4502       \tag_resume:n {keyanspic}
4503     }
4504   }
4505 }

```

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-ans*` and `wrap-opt` keys.

```

4506 \cs_new_protected:Npn \__enumext_anspic_label:nn #1 #2
4507 {
4508   \makebox[ \l__enumext_anspic_mini_width_dim ][ c ]
4509   {
4510     \bool_if:nTF { #1 }
4511     {
4512       \bool_set_true:N \l__enumext_item_wrap_key_bool
4513       \bool_set_true:N \l__enumext_wrap_label_v_bool
4514       \__enumext_keyans_save_item_opt:n { #2 }
4515       \__enumext_keyans_addto_prop:n { #2 }
4516       \__enumext_keyans_store_ref:
4517       \__enumext_keyans_addto_seq:n { #2 }
4518       \int_gincr:N \g__enumext_check_starred_cmd_int
4519       \__enumext_keyans_show_ans:
4520       \__enumext_keyans_show_pos:
4521       \makebox[ \l__enumext_labelwidth_v_dim ][ c ]
4522       {
4523         \tl_use:N \l__enumext_label_font_style_v_tl
4524         \__enumext_keyans_wrapper_label:n { \l__enumext_label_vi_tl }
4525       }
4526       \skip_horizontal:n { \l__enumext_labelsep_v_dim }
4527       \__enumext_keyans_show_item_opt:
4528     }
4529     {
4530       \bool_set_false:N \l__enumext_item_wrap_key_bool
4531       \tl_use:N \l__enumext_label_font_style_v_tl
4532       \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl }
4533     }
4534   }
4535 }

```

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.

```

4536 \cs_new_protected:Npn \__enumext_anspic_label_pos:nnn #1 #2 #3
4537 {
4538   \stepcounter { enumXvi }
4539   \__enumext_anspic_body_dim:n { #3 }
4540   \bool_if:NTF \l__enumext_anspic_label_above_bool
4541   {
4542     \__enumext_anspic_label:nn { #1 } { #2 }
4543   }
4544   {
4545     \raisebox
4546     {
4547       -\dim_eval:n
4548       {
4549         \l__enumext_anspic_label_htdp_dim
4550         + \l__enumext_anspic_body_htdp_dim
4551         + \box_dp:N \strutbox
4552         + \l__enumext_anspic_label_sep_skip
4553       }
4554     }
4555     [ opt ] [ opt ]
4556     {
4557       \__enumext_anspic_label:nn { #1 } { #2 }
4558     }
4559   }
4560 }
4561 %

```

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.

```

4562 \cs_new_protected:Nn \__enumext_anspic_args:nnn
4563 {
4564   \__enumext_anspic_start_list_tag:
4565   \__enumext_anspic_label_pos:nnn { #1 } { #2 } { #3 }
4566   \__enumext_anspic_stop_start_list_tag:
4567   \bool_if:NTF \l__enumext_anspic_label_above_bool
4568   {
4569     \[\l__enumext_anspic_label_sep_skip] #3
4570   }
4571   {
4572     \[ #3
4573   }
4574   \__enumext_anspic_stop_list_tag:
4575 }

```

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`.

```

4576 \cs_new_protected:Nn \__enumext_anspic_print:n
4577 {
4578   \clist_map_function:nN { #1 } \__enumext_anspic_row:n
4579 }
4580 \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.

```

4581 \cs_new_protected:Nn \__enumext_anspic_row:n
4582 {
4583   \dim_set:Nn \l__enumext_anspic_mini_width_dim { \linewidth / #1 }
4584   \int_set:Nn \l__enumext_anspic_above_int { \l__enumext_anspic_below_int }
4585   \int_set:Nn \l__enumext_anspic_below_int { \l__enumext_anspic_above_int + #1 }
4586   \int_step_inline:nnn
4587   { \l__enumext_anspic_above_int + 1 }
4588   { \l__enumext_anspic_below_int }
4589   {
4590     \IfDocumentMetadataT
4591     {
4592       \tag_suspend:n {minipage}
4593     }
4594     \begin{minipage}[ \l__enumext_anspic_mini_pos_str ]{ \l__enumext_anspic_mini_width_dim }
4595     \centering

```



```

4596         \seq_item:Nn \l__enumext_anspic_args_seq { ##1 }
4597     \end{minipage}
4598     \IfDocumentMetadataT
4599     {
4600         \tag_resume:n {minipage}
4601     }
4602 }
4603 \par
4604 }

```

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*.

```

4605 \cs_new_protected:Nn \__enumext_anspic_exec:
4606 {
4607     \tl_if_empty:NTF \l__enumext_anspic_layout_style_tl
4608     {
4609         \__enumext_anspic_print:e { \seq_count:N \l__enumext_anspic_args_seq }
4610     }
4611     {
4612         \__enumext_anspic_print:V \l__enumext_anspic_layout_style_tl
4613     }
4614 }

```

(End of definition for `\anspic` and others. This function is documented on page 17.)

13.45 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.

13.45.1 Functions for item box width

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

```

\__enumext_starred_columns_set_vii:
\__enumext_starred_columns_set_viii:
4615 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
4616 {
4617     \dim_compare:nNt { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
4618     {
4619         \dim_set:Nn \l__enumext_columns_sep_vii_dim
4620         {
4621             ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
4622             / \l__enumext_columns_vii_int
4623         }
4624     }
4625     \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - 1 }
4626     \dim_set:Nn \l__enumext_item_width_vii_dim
4627     {
4628         ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
4629         / \l__enumext_columns_vii_int
4630         - \l__enumext_labelwidth_vii_dim
4631         - \l__enumext_labelsep_vii_dim
4632     }

```

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

```

4633 \dim_compare:nNtT { \l__enumext_rightmargin_vii_dim } > { \c_zero_dim }
4634 {
4635   \dim_sub:Nn \l__enumext_item_width_vii_dim
4636   {
4637     ( \l__enumext_rightmargin_vii_dim * \l__enumext_tmpa_vii_int )
4638     / \l__enumext_columns_vii_int
4639   }
4640   \dim_add:Nn \l__enumext_columns_sep_vii_dim
4641   {
4642     \l__enumext_rightmargin_vii_dim
4643   }
4644 }
4645 }

```

Same implementation for the `keyans*` environment.

```

4646 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
4647 {
4648   \dim_compare:nNtT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
4649   {
4650     \dim_set:Nn \l__enumext_columns_sep_viii_dim
4651     {
4652       ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
4653       / \l__enumext_columns_viii_int
4654     }
4655   }
4656   \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - 1 }
4657   \dim_set:Nn \l__enumext_item_width_viii_dim
4658   {
4659     ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
4660     / \l__enumext_columns_viii_int
4661     - \l__enumext_labelwidth_viii_dim
4662     - \l__enumext_labelsep_viii_dim
4663   }
4664   \dim_compare:nNtT { \l__enumext_rightmargin_viii_dim } > { \c_zero_dim }
4665   {
4666     \dim_sub:Nn \l__enumext_item_width_viii_dim
4667     {
4668       ( \l__enumext_rightmargin_viii_dim * \l__enumext_tmpa_vii_int )
4669       / \l__enumext_columns_viii_int
4670     }
4671     \dim_add:Nn \l__enumext_columns_sep_viii_dim
4672     {
4673       \l__enumext_rightmargin_viii_dim
4674     }
4675   }
4676 }

```

(End of definition for `__enumext_starred_columns_set_vii:` and `__enumext_starred_columns_set_viii:`.)

13.45.2 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.

```

4677 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
4678 {
4679   \int_set:Nn \l__enumext_joined_item_vii_int {#1}
4680   \int_compare:nNtT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
4681   {
4682     \msg_warning:nnee { enumext } { item-joined }
4683     { \int_use:N \l__enumext_joined_item_vii_int }
4684     { \int_use:N \l__enumext_columns_vii_int }
4685     \int_set:Nn \l__enumext_joined_item_vii_int
4686     {
4687       \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4688     }
4689   }
4690   \int_compare:nNtT
4691   { \l__enumext_joined_item_vii_int }
4692   >
4693   { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }

```

```

4694     {
4695         \msg_warning:nnee { enumext } { item-joined-columns }
4696         { \int_use:N \l__enumext_joined_item_vii_int }
4697         {
4698             \int_eval:n
4699                 { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4700         }
4701         \int_set:Nn \l__enumext_joined_item_vii_int
4702         {
4703             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4704         }
4705     }
4706     \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { 1 }
4707     {
4708         \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
4709         \int_decr:N \l__enumext_joined_item_aux_vii_int
4710         \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
4711         \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
4712         \dim_set:Nn \l__enumext_joined_width_vii_dim
4713         {
4714             \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
4715             + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
4716               + \l__enumext_columns_sep_vii_dim
4717             ) * \l__enumext_joined_item_aux_vii_int
4718         }
4719         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
4720     }
4721     {
4722         \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
4723         \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
4724     }
4725 }

```

Same implementation for the [keyans*](#) environment.

```

4726 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
4727 {
4728     \int_set:Nn \l__enumext_joined_item_viii_int {#1}
4729     \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
4730     {
4731         \msg_warning:nnee { enumext } { item-joined }
4732         { \int_use:N \l__enumext_joined_item_viii_int }
4733         { \int_use:N \l__enumext_columns_viii_int }
4734         \int_set:Nn \l__enumext_joined_item_viii_int
4735         {
4736             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4737         }
4738     }
4739     \int_compare:nNnT
4740     { \l__enumext_joined_item_viii_int }
4741     >
4742     { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4743     {
4744         \msg_warning:nnee { enumext } { item-joined-columns }
4745         { \int_use:N \l__enumext_joined_item_viii_int }
4746         {
4747             \int_eval:n
4748                 { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4749         }
4750         \int_set:Nn \l__enumext_joined_item_viii_int
4751         {
4752             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4753         }
4754     }
4755     \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { 1 }
4756     {
4757         \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
4758         \int_decr:N \l__enumext_joined_item_aux_viii_int
4759         \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
4760         \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
4761         \dim_set:Nn \l__enumext_joined_width_viii_dim
4762         {
4763             \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int

```

```

4764         + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
4765           + \l__enumext_columns_sep_viii_dim
4766           )*\l__enumext_joined_item_aux_viii_int
4767     }
4768     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
4769   }
4770   {
4771     \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
4772     \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
4773   }
4774 }

```

(End of definition for \l__enumext_starred_joined_item_vii:n and \l__enumext_starred_joined_item_viii:n)

13.45.3 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`.

```

4775 \cs_new_protected:Nn \__enumext_start_mini_vii:
4776 {
4777   \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
4778   {
4779     \dim_set:Nn \l__enumext_minipage_left_vii_dim
4780     {
4781       \linewidth
4782       - \l__enumext_minipage_right_vii_dim
4783       - \l__enumext_minipage_hsep_vii_dim
4784     }
4785     \bool_set_true:N \l__enumext_minipage_active_vii_bool
4786     \dim_gset_eq:NN
4787       \g__enumext_minipage_right_vii_dim
4788       \l__enumext_minipage_right_vii_dim
4789     \__enumext_mini_addvspace_vii:
4790     \nointerlineskip\noindent
4791     \__enumext_mini_page{ \l__enumext_minipage_left_vii_dim }
4792   }
4793 }

```

The function `__enumext_stop_mini_vii:` closes the `__enumext_mini_page` environment on the “left side”, applies `\hfill` and set the variable `\g__enumext_minipage_active_vii_bool` to “true” which will be used in the function `__enumext_after_env:n` to execute the `minipage` on the “right side”. 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 (§13.46).

```

4794 \cs_new_protected:Nn \__enumext_stop_mini_vii:
4795 {
4796   \bool_if:NTF \l__enumext_minipage_active_vii_bool
4797   {
4798     \__enumext_stop_list:
4799     \__enumext_stop_store_level_vii:
4800     \IfDocumentMetadataT { \tag_resume:n {enumext*} }
4801     \end__enumext_mini_page
4802     \hfill
4803     \bool_gset_true:N \g__enumext_minipage_active_vii_bool
4804   }
4805   {
4806     \__enumext_stop_list:
4807     \__enumext_stop_store_level_vii:
4808   }
4809 }

```

(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.

```

4810 \__enumext_after_env:n {enumext*}
4811 {

```

```

4812 \bool_if:NT \g__enumext_minipage_active_vii_bool
4813 {
4814   \__enumext_minipage:w [ t ] { \g__enumext_minipage_right_vii_dim }
4815   \legacy_if_gset_false:n { @minipage }
4816   \skip_vertical:N \c_zero_skip
4817   \par\addvspace { \g__enumext_minipage_right_skip }
4818   \bool_if:NF \g__enumext_minipage_center_vii_bool
4819   {
4820     \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
4821     {
4822       \centering
4823     }
4824   }
4825   \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
4826   {
4827     \tl_use:N \g__enumext_miniright_code_vii_tl
4828   }
4829   \box_use_drop:N \l__enumext_miniright_code_vii_box
4830   \skip_vertical:N \c_zero_skip
4831   \__enumext_endminipage:
4832   \par\addvspace{ \g__enumext_minipage_after_skip }
4833 }
4834 \bool_gset_false:N \g__enumext_minipage_active_vii_bool
4835 \bool_gset_true:N \g__enumext_minipage_center_vii_bool
4836 \tl_gclear:N \g__enumext_miniright_code_vii_tl
4837 \dim_gzero:N \g__enumext_minipage_right_vii_dim
4838 \bool_gset_false:N \g__enumext_starred_bool
4839 }

```

```

\__enumext_start_mini_viii:
\__enumext_stop_mini_viii:

```

The implementation of the `mini-env`, `mini-right` and `mini-right*` keys is identical to the one used in the `enumext*` environment.

```

4840 \cs_new_protected:Nn \__enumext_start_mini_viii:
4841 {
4842   \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
4843   {
4844     \dim_set:Nn \l__enumext_minipage_left_viii_dim
4845     {
4846       \linewidth
4847       - \l__enumext_minipage_right_viii_dim
4848       - \l__enumext_minipage_hsep_viii_dim
4849     }
4850     \bool_set_true:N \l__enumext_minipage_active_viii_bool
4851     \dim_gset_eq:NN
4852     \g__enumext_minipage_right_viii_dim
4853     \l__enumext_minipage_right_viii_dim
4854     \__enumext_mini_addvspace_viii:
4855     \nointerlineskip\noindent
4856     \__enumext_mini_page{ \l__enumext_minipage_left_viii_dim }
4857   }
4858 }
4859 \cs_new_protected:Nn \__enumext_stop_mini_viii:
4860 {
4861   \bool_if:NTF \l__enumext_minipage_active_viii_bool
4862   {
4863     \__enumext_stop_list:
4864     \IfDocumentMetadataTF { \tag_resume:n {keyans*} } { }
4865     \end__enumext_mini_page
4866     \hfill
4867     \bool_gset_true:N \g__enumext_minipage_active_viii_bool
4868   }
4869   {
4870     \__enumext_stop_list:
4871   }
4872 }
4873 \__enumext_after_env:nn {keyans*}
4874 {
4875   \bool_if:NT \g__enumext_minipage_active_viii_bool
4876   {
4877     \__enumext_mini_page{ \g__enumext_minipage_right_viii_dim }
4878     \par\addvspace { \g__enumext_minipage_right_skip }
4879     \bool_if:NF \g__enumext_minipage_center_viii_bool

```

```

4880         {
4881             \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
4882             {
4883                 \centering
4884             }
4885         }
4886         \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
4887         {
4888             \tl_use:N \g__enumext_miniright_code_viii_tl
4889         }
4890         \box_use_drop:N \l__enumext_miniright_code_viii_box
4891         \end__enumext_mini_page
4892         \par\addvspace{ \g__enumext_minipage_after_skip }
4893     }
4894     \bool_gset_false:N \g__enumext_minipage_active_viii_bool
4895     \bool_gset_true:N \g__enumext_minipage_center_viii_bool
4896     \tl_gclear:N \g__enumext_miniright_code_viii_tl
4897     \dim_gzero:N \g__enumext_minipage_right_viii_dim
4898 }

```

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

13.46 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.

```

4899 \NewDocumentEnvironment{enumext*}{ o }
4900 {
4901     \__enumext_safe_exec_vii:
4902     \__enumext_parse_keys_vii:n {#1}
4903     \__enumext_before_list_vii:
4904     \__enumext_start_store_level_vii:
4905     \__enumext_start_list:nn { }
4906     {
4907         \__enumext_list_arg_two_vii:
4908         \__enumext_before_keys_exec_vii:
4909     }
4910     \setcounter { enumXvii } { \int_eval:n { \int_use:c { \l__enumext_start_vii_int } - 1 } }
4911     \IfDocumentMetadataT { \tag_suspend:n {enumext*} }
4912     \__enumext_starred_columns_set_vii:
4913     \item[] \scan_stop:
4914     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_first_item_tmp_vii:
4915     \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
4916     \ignorespaces
4917 }
4918 {
4919     \IfDocumentMetadataT { \tag_struct_end:n {tag=text-unit} }
4920     \__enumext_stop_item_tmp_vii:
4921     \__enumext_remove_extra_parsep_vii:
4922     \__enumext_after_list_vii:
4923 }

```

(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.

```

4924 \cs_new_protected:Nn \__enumext_safe_exec_vii:
4925 {
4926     \__enumext_is_not_nested:
4927     \__enumext_internal_mini_page:
4928     \int_incr:N \l__enumext_level_h_int
4929     \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
4930     {

```

```

4931         \msg_error:nn { enumext } { nested }
4932     }
4933     \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
4934     {
4935         \msg_error:nnn { enumext } { nested-horizontal } { keyans*}
4936     }
4937     \bool_set_true:N \l__enumext_starred_bool
4938     \bool_set_false:N \l__enumext_standar_bool
4939     \__enumext_is_on_first_level:
4940 }

```

(End of definition for __enumext_safe_exec_vii:.)

__enumext_parse_keys_vii:n First we will clear the variable \l__enumext_series_name_str used by the key `series`, process the environment $\langle key = val \rangle$ 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 $\langle keys \rangle$ to pass them to the storage *sequence* if the key `save-key` is not active.

```

4941 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
4942 {
4943     \tl_if_novalue:nF {#1}
4944     {
4945         \str_clear:N \l__enumext_series_name_str
4946         \keys_set:nn { enumext / enumext* } {#1}
4947         \bool_if:NF \l__enumext_print_keyans_cmd_bool
4948         {
4949             \__enumext_parse_series:n {#1}
4950         }
4951         \__enumext_store_active_keys_vii:n {#1}
4952     }
4953 }

```

(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*`.

```

4954 \cs_new_protected:Nn \__enumext_before_list_vii:
4955 {
4956     \__enumext_vspace_above_vii:
4957     \__enumext_check_ans_active:
4958     \__enumext_before_args_exec_vii:
4959     \__enumext_start_mini_vii:
4960 }

```

(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: (§13.45.3) 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 \l__enumext_starred_bool to false and call the __enumext_resume_save_counter: function used by the `series`, `resume` and `resume*` keys.

```

4961 \cs_new_protected:Nn \__enumext_after_list_vii:
4962 {
4963     \__enumext_stop_mini_vii:
4964     \__enumext_after_stop_list_vii:
4965     \__enumext_check_ans_key_hook:
4966     \__enumext_vspace_below_vii:
4967     \bool_set_false:N \l__enumext_starred_bool
4968     \bool_if:NF \l__enumext_print_keyans_cmd_bool
4969     {
4970         \__enumext_starred_save_counter:
4971     }
4972 }

```

(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`.

```

4973 \cs_new_protected:Nn \__enumext_start_store_level_vii:
4974 {
4975   \bool_if:NT \__enumext_store_active_bool
4976   {
4977     \int_compare:nNnT { \__enumext_level_int } > { 0 }
4978     {
4979       \__enumext_store_level_open_vii:
4980     }
4981   }
4982 }
4983 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
4984 {
4985   \bool_if:NT \__enumext_store_active_bool
4986   {
4987     \int_compare:nNnT { \__enumext_level_int } > { 0 }
4988     {
4989       \__enumext_store_level_close_vii:
4990     }
4991   }
4992 }

```

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

13.46.1 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.

```

4993 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_vii:
4994 {
4995   \skip_horizontal:n
4996   {
4997     -\__enumext_labelwidth_vii_dim - \__enumext_labelsep_vii_dim
4998   }
4999   \ignorespaces
5000 }

```

(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`.

```
\__enumext_starred_item_vii_aux_i:w
```

```
\__enumext_starred_item_vii_aux_ii:w
```

```
\__enumext_starred_item_vii_aux_iii:w
```

```

5001 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
5002 {
5003   \__enumext_stop_item_tmp_vii:
5004   \int_incr:N \__enumext_item_column_pos_vii_int
5005   \int_gincr:N \g__enumext_item_count_all_vii_int
5006   \__enumext_item_peek_args_vii:
5007 }

```

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).

```

5008 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
5009 {
5010   \peek_meaning:NTF (
5011     { \__enumext_joined_item_vii:w }
5012     { \__enumext_joined_item_vii:w (1) }
5013   )

```

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`.

```

5014 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
5015 {

```

```

5016     \__enumext_starred_joined_item_vii:n {#1}
5017     \peek_meaning_remove:NTF *
5018     { \__enumext_starred_item_vii:w }
5019     { \__enumext_standar_item_vii:w }
5020 }

```

The function `__enumext_standar_item_vii:w` will first look for the argument “[”, if present it will set the state of the variable `__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 [__enumext_label_vii_tl]`.

```

5021 \cs_new_protected:Npn \__enumext_standar_item_vii:w
5022 {
5023     \bool_set_false:N \l__enumext_item_starred_vii_bool
5024     \peek_meaning:NTF [
5025     {
5026         \bool_set_eq:NN \l__enumext_wrap_label_vii_bool \l__enumext_wrap_label_opt_vii_bool
5027         \__enumext_start_item_vii:w
5028     }
5029     {
5030         \bool_set_true:N \l__enumext_wrap_label_vii_bool
5031         \legacy_if_set_true:n { @noitemarg }
5032         \__enumext_start_item_vii:w [ \__enumext_label_vii_tl ] \ignorespaces
5033     }
5034 }

```

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⟩]`.

```

5035 \cs_new_protected:Npn \__enumext_starred_item_vii:w
5036 {
5037     \bool_set_true:N \l__enumext_item_starred_vii_bool
5038     \bool_set_true:N \l__enumext_wrap_label_vii_bool
5039     \peek_meaning:NTF [
5040     { \__enumext_starred_item_vii_aux_i:w }
5041     { \__enumext_starred_item_vii_aux_ii:w }
5042 }
5043 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w {#1}
5044 {
5045     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
5046     \__enumext_starred_item_vii_aux_ii:w
5047 }
5048 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
5049 {
5050     \peek_meaning:NTF [
5051     { \__enumext_starred_item_vii_aux_iii:w }
5052     {
5053         \dim_set_eq:NN \l__enumext_item_symbol_sep_vii_dim \l__enumext_labelsep_vii_dim
5054         \legacy_if_set_true:n { @noitemarg }
5055         \__enumext_start_item_vii:w [ \__enumext_label_vii_tl ] \ignorespaces
5056     }
5057 }
5058 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w {#1}
5059 {
5060     \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
5061     \legacy_if_set_true:n { @noitemarg }
5062     \__enumext_start_item_vii:w [ \__enumext_label_vii_tl ] \ignorespaces
5063 }

```

(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*[⟨symbol⟩][⟨offset⟩]` 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`

```

5064 \cs_new_protected_nopar:Npn \__enumext_fake_make_label_vii:n #1
5065 {
5066   \legacy_if:nT { @noitemarg }
5067   {
5068     \legacy_if_set_false:n { @noitemarg }
5069     \legacy_if:nT { @nmbrrlist }
5070     {
5071       \IfDocumentMetadataT
5072       {
5073         \bool_if:NT \__enumext_hyperref_bool
5074         {
5075           \legacy_if_set_true:n { @hyper@item }
5076         }
5077       }
5078       \refstepcounter{enumXvii}
5079       \bool_if:NT \__enumext_check_answers_bool
5080       {
5081         \int_gincr:N \g__enumext_item_number_int
5082         \bool_set_true:N \__enumext_item_number_bool
5083       }
5084     }
5085   }
5086   \bool_if:NT \__enumext_item_starred_vii_bool
5087   {
5088     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
5089     {
5090       \tl_gset_eq:NN
5091       \g__enumext_item_symbol_aux_vii_tl \__enumext_item_symbol_vii_tl
5092     }
5093     \mode_leave_vertical:
5094     \skip_horizontal:n { -\__enumext_item_symbol_sep_vii_dim }
5095     \hbox_overlap_left:n { \g__enumext_item_symbol_aux_vii_tl }
5096     \skip_horizontal:N \__enumext_item_symbol_sep_vii_dim
5097     \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
5098   }
5099   \makebox[ \__enumext_labelwidth_vii_dim ][ \__enumext_align_label_vii_str ]
5100   {
5101     \tl_use:N \__enumext_label_font_style_vii_tl
5102     \bool_if:NTF \__enumext_wrap_label_vii_bool
5103     {
5104       \__enumext_wrapper_label_vii:n {#1}
5105     }
5106     { #1 }
5107   }
5108   \skip_horizontal:N \__enumext_labelsep_vii_dim \ignorespaces
5109 }

```

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

13.46.2 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`.

```

5110 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
5111 {
5112   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
5113   \hbox_set_to_wd:Nnw \__enumext_item_text_vii_box
5114   {
5115     \__enumext_joined_width_vii_dim
5116     + \__enumext_labelwidth_vii_dim
5117     + \__enumext_labelsep_vii_dim
5118   }

```

Redefine the `\footnote` command.

```

5119   \__enumext_renew_footnote_starred:

```

Now we insert our *sockets* for *tagging* PDF support and run `\item`.

```
5120 \__enumext_start_list_tag:n {enumext*}
5121 \__enumext_fake_make_label_vii:n {#1}
5122 \__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.

```
5123 \__enumext_minipage:w [ t ]{ \__enumext_joined_width_vii_dim }
5124 \dim_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
5125 \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
5126 \__enumext_unskip_unkern:
5127 \__enumext_unskip_unkern:
5128 \skip_horizontal:n { -\l__enumext_listparindent_vii_dim } \ignorespaces
5129 \tl_use:N \l__enumext_fake_item_indent_vii_tl
5130 \tl_use:N \l__enumext_after_list_args_vii_tl
5131 }
```

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*.

```
5132 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
5133 {
5134     \__enumext_endminipage:
5135     \__enumext_stop_list_tag:n {enumext*}
5136     \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*.

```
5137 \int_set:Nn \hbadness { 10000 }
5138 \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`.

```
5139 \int_compare:nNnTF
5140 { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
5141 {
5142     \par\noindent
5143     \int_zero:N \l__enumext_item_column_pos_vii_int
5144 }
5145 {
5146     \skip_horizontal:N \l__enumext_columns_sep_vii_dim
5147 }
5148 }
```

(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*).

```
5149 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
5150 {
5151     \int_compare:nNnT
5152     {
5153         \int_mod:nn
5154         { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
5155     }
5156     =
5157     { 0 }
5158     {
5159         \para_end:
5160         \skip_vertical:n { -\l__enumext_itemsep_vii_skip }
5161         \skip_vertical:N \c_zero_skip
5162         \int_gzero:N \g__enumext_item_count_all_vii_int
5163     }
5164 }
```

(End of definition for `__enumext_remove_extra_parsep_viii:`)

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`.

```
5165 \__enumext_after_env:nn {enumext*}
5166 {
5167     \__enumext_execute_after_env:
5168 }
```

13.47 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.

```
5169 \NewDocumentEnvironment{keyans*}{ o }
5170 {
5171     \__enumext_safe_exec_viii:
5172     \__enumext_parse_keys_viii:n {#1}
5173     \__enumext_before_list_viii:
5174     \__enumext_start_list:nn { }
5175     {
5176         \__enumext_list_arg_two_viii:
5177         \__enumext_before_keys_exec_viii:
5178     }
5179     \setcounter { enumXviii } { \int_eval:n { \int_use:c { l__enumext_start_viii_int } - 1 } }
5180     \IfDocumentMetadataT { \tag_suspend:n {keyans*} }
5181     \__enumext_starred_columns_set_viii:
5182     \item[] \scan_stop:
5183     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_first_item_tmp_viii:
5184     \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
5185     \ignorespaces
5186 }
5187 {
5188     \IfDocumentMetadataT { \tag_struct_end:n {tag=text-unit} }
5189     \__enumext_stop_item_tmp_viii:
5190     \__enumext_remove_extra_parsep_viii:
5191     \__enumext_check_starred_cmd:n { item }
5192     \__enumext_after_list_viii:
5193 }
```

(End of definition for `keyans*`. This function is documented on page 16.)

`__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.

```
5194 \cs_new_protected:Nn \__enumext_safe_exec_viii:
5195 {
5196     \bool_if:NF \l__enumext_store_active_bool
5197     {
5198         \msg_error:nnnn { enumext } { wrong-place } { keyans* } { save-ans }
5199     }
5200     \int_incr:N \l__enumext_keyans_level_h_int
5201     \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
5202     {
5203         \msg_error:nn { enumext } { nested }
5204     }
5205     \__enumext_keyans_name_and_start:
5206     \bool_if:NT \l__enumext_starred_bool
5207     {
5208         \msg_error:nnn { enumext } { nested-horizontal } { enumext* }
5209     }
5210     \bool_set_true:N \l__enumext_starred_bool
5211     % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
5212     \bool_set_false:N \l__enumext_store_active_bool
5213     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
5214     {
5215         \msg_error:nn { enumext } { keyans-wrong-level }
5216     }
5217 }
```

(End of definition for `__enumext_safe_exec_viii:`)

```
\__enumext_parse_keys_viii:n Parse [ $\langle key = val \rangle$ ] for keyans*.
5218 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
5219 {
5220   \tl_if_novalue:nF {#1}
5221   {
5222     \keys_set:nn { enumext / keyans* } {#1}
5223   }
5224 }
```

(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 $\{\langle code \rangle\}$ defined by the `before*` key if it is active, the call the function `__enumext_start_mini_viii:` handle by `mini-env`.

```
5225 \cs_new_protected:Nn \__enumext_before_list_viii:
5226 {
5227   \__enumext_vspace_above_viii:
5228   \__enumext_before_args_exec_viii:
5229   \__enumext_start_mini_viii:
5230 }
```

(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 $\{\langle code \rangle\}$ handled by the `after` key together with the *vertical space* handled by the `below` key if they are present.

```
5231 \cs_new_protected:Nn \__enumext_after_list_viii:
5232 {
5233   \__enumext_stop_mini_viii:
5234   \__enumext_after_stop_list_viii:
5235   \__enumext_vspace_below_viii:
5236 }
```

(End of definition for `__enumext_after_list_viii:`.)

13.47.1 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* ($\langle number \rangle$) which works in the same way as in the `enumext*` environment. In simple terms we want to store the $\langle label \rangle$ next to the $[\langle content \rangle]$ if it is present in the *sequence* and *prop list* defined by `save-ans` key for `\item*`, `\item*[\langle content \rangle]`, `\item(\langle number \rangle)*` and `\item(\langle number \rangle)*[\langle content \rangle]` 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.

```
5237 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_viii:
5238 {
5239   \skip_horizontal:n
5240   {
5241     -\__enumext_labelwidth_viii_dim - \__enumext_labelsep_viii_dim
5242   }
5243   \ignorespaces
5244 }
```

(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 `__enumext_item_column_pos_viii_int` that will count the item’s by rows and the value of `\g__enumext_item_count_all_viii_int` that will count the total of item’s in the environment. `__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`.

```
5245 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
5246 {
5247   \__enumext_stop_item_tmp_viii:
5248   \int_incr:N \__enumext_item_column_pos_viii_int
5249   \int_gincr:N \g__enumext_item_count_all_viii_int
5250   \__enumext_item_peek_args_viii:
5251 }
```

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).

```
5252 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
5253 {
5254   \peek_meaning:NTF (
5255     { \__enumext_joined_item_viii:w }
5256     { \__enumext_joined_item_viii:w (1) }
5257   }
```

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`.

```
5258 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
5259 {
5260   \__enumext_starred_joined_item_viii:n {#1}
5261   \peek_meaning_remove:NTF *
5262     { \__enumext_starred_item_viii:w }
5263     { \__enumext_standar_item_viii:w }
5264 }
```

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]`.

```
5265 \cs_new_protected:Npn \__enumext_standar_item_viii:w
5266 {
5267   \bool_set_false:N \l__enumext_item_starred_viii_bool
5268   \bool_set_false:N \l__enumext_item_wrap_key_bool
5269   \peek_meaning:NTF [
5270     {
5271       \bool_set_eq:NN \l__enumext_wrap_label_viii_bool \l__enumext_wrap_label_opt_viii_bool
5272       \__enumext_start_item_viii:w
5273     }
5274     {
5275       \bool_set_true:N \l__enumext_wrap_label_viii_bool
5276       \legacy_if_set_true:n { @noitemarg }
5277       \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ] \ignorespaces
5278     }
5279   }
```

(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* [<content>]`.

```
5280 \cs_new_protected:Npn \__enumext_starred_item_viii:w
5281 {
5282   \bool_set_true:N \l__enumext_item_starred_viii_bool
5283   \bool_set_true:N \l__enumext_item_wrap_key_bool
5284   \bool_set_true:N \l__enumext_wrap_label_viii_bool
5285   \peek_meaning:NTF [
5286     { \__enumext_starred_item_viii_aux_i:w }
5287     { \__enumext_starred_item_viii_aux_ii:w }
5288   }
```

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`.

```
5289 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
5290 {
5291   \tl_clear:N \l__enumext_store_current_label_tl
5292   \tl_if_no_value:nF { #1 }
5293   {
5294     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_viii_tl
5295     {
5296       \tl_put_right:NV \l__enumext_store_current_label_tl \l__enumext_store_keyans_item_opt.
```



```

5297         \tl_put_right:Nn \l__enumext_store_current_label_tl { #1 }
5298     }
5299     \tl_set:Nn \l__enumext_store_current_opt_arg_tl { #1 }
5300 }
5301 \__enumext_starred_item_viii_aux_ii:w
5302 }
5303 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
5304 {
5305     \legacy_if_set_true:n { @noitemarg }
5306     \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ] \ignorespaces
5307 }

```

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

```

5308 \cs_new_protected:Nn \__enumext_keyans_starred_item_star:
5309 {
5310     \tl_put_left:Ne \l__enumext_store_current_label_tl { \l__enumext_label_viii_tl }
5311     \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
5312     \__enumext_keyans_store_ref:
5313     \tl_put_left:Nn \l__enumext_store_current_label_tl { \item }
5314     \__enumext_keyans_addto_seq_link:
5315     \int_gincr:N \g__enumext_check_starred_cmd_int
5316     \dim_compare:nNt { \l__enumext_mark_sym_sep_viii_dim } = { \c_zero_dim }
5317     {
5318         \dim_set:Nn \l__enumext_mark_sym_sep_viii_dim { \l__enumext_labelsep_viii_dim }
5319     }
5320     \bool_if:NT \l__enumext_show_answer_bool
5321     {
5322         \tl_set_eq:NN \l__enumext_mark_answer_sym_tl \l__enumext_mark_answer_sym_viii_tl
5323         \str_set_eq:NN \l__enumext_mark_position_str \l__enumext_mark_position_viii_str
5324         \__enumext_print_keyans_box:NN
5325         \l__enumext_labelwidth_viii_dim \l__enumext_mark_sym_sep_viii_dim
5326     }
5327     \bool_if:NT \l__enumext_show_position_bool
5328     {
5329         \tl_set:Ne \l__enumext_mark_answer_sym_tl
5330         {
5331             \group_begin:
5332             \exp_not:N \normalfont
5333             \exp_not:N \footnotesize [ \int_eval:n
5334             {
5335                 \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
5336             }
5337             ]
5338             \group_end:
5339         }
5340         \str_set_eq:NN \l__enumext_mark_position_str \l__enumext_mark_position_viii_str
5341         \__enumext_print_keyans_box:NN
5342         \l__enumext_labelwidth_viii_dim \l__enumext_mark_sym_sep_viii_dim
5343     }
5344 }

```

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

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

```

\__enumext_keyans_wrapper_label_viii:n
\__enumext_fake_make_label_viii:n
5345 \cs_new_protected:Npn \__enumext_keyans_wrapper_label_viii:n #1
5346 {
5347     \bool_lazy_all:nT
5348     {
5349         { \bool_if_p:N \l__enumext_wrap_label_viii_bool }
5350         { \bool_if_p:N \l__enumext_show_answer_bool }
5351         { \bool_if_p:N \l__enumext_item_wrap_key_bool }
5352         { \cs_if_exist_p:N \__enumext_keyans_wrapper_item_viii:n }
5353     }
5354     {
5355         \cs_set_eq:NN
5356         \__enumext_wrapper_label_viii:n \__enumext_keyans_wrapper_item_viii:n
5357     }
5358     \bool_if:NTF \l__enumext_wrap_label_viii_bool

```

```

5359     {
5360         \__enumext_wrapper_label_viii:n {#1}
5361     }
5362     { #1 }
5363 }
5364 \cs_new_protected_nopar:Npn \__enumext_fake_make_label_viii:n #1
5365 {
5366     \legacy_if:nT { @noitemarg }
5367     {
5368         \legacy_if_set_false:n { @noitemarg }
5369         \legacy_if:nT { @nmbrlist }
5370         {
5371             \refstepcounter{enumXviii}
5372         }
5373     }
5374     \bool_if:NT \__enumext_item_starred_viii_bool
5375     {
5376         \__enumext_keyans_starred_item_star:
5377     }
5378     \makebox[ \__enumext_labelwidth_viii_dim ][ \__enumext_align_label_viii_str ]
5379     {
5380         \tl_use:N \__enumext_label_font_style_viii_tl
5381         \__enumext_keyans_wrapper_label_viii:n {#1}
5382     }
5383     \skip_horizontal:N \__enumext_labelsep_viii_dim \ignorespaces
5384 }

```

(End of definition for `__enumext_keyans_wrapper_label_viii:n` and `__enumext_fake_make_label_viii:n`)

13.47.2 Real definition of `\item` in `keyans*`

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

```

\__enumext_start_item_viii:w \__enumext_stop_item_viii:
5385 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
5386 {
5387     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
5388     \hbox_set_to_wd:Nnw \__enumext_item_text_viii_box
5389     {
5390         \__enumext_joined_width_viii_dim
5391         + \__enumext_labelwidth_viii_dim
5392         + \__enumext_labelsep_viii_dim
5393     }
5394     \__enumext_renew_footnote_starred:
5395     \__enumext_start_list_tag:n {keyans*}
5396     \__enumext_fake_make_label_viii:n {#1}
5397     \__enumext_stop_start_list_tag:
5398     \__enumext_minipage:w [ t ]{ \__enumext_joined_width_viii_dim }
5399     \dim_set_eq:NN \parindent \__enumext_listparindent_viii_dim
5400     \skip_set_eq:NN \parskip \__enumext_parsep_viii_skip
5401     \__enumext_unskip_unkern:
5402     \__enumext_unskip_unkern:
5403     \skip_horizontal:n { -\__enumext_listparindent_viii_dim } \ignorespaces
5404     \tl_use:N \__enumext_fake_item_indent_viii_tl
5405     \bool_if:NT \__enumext_item_starred_viii_bool
5406     {
5407         \__enumext_keyans_show_item_opt_viii:
5408     }
5409     \tl_use:N \__enumext_after_list_args_viii_tl
5410 }
5411 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
5412 {
5413     \__enumext_endminipage:
5414     \__enumext_stop_list_tag:n {keyans*}
5415     \hbox_set_end:
5416     \int_set:Nn \hbadness { 10000 }
5417     \box_use_drop:N \__enumext_item_text_viii_box
5418     \int_compare:nNnTF
5419     { \__enumext_item_column_pos_viii_int } = { \__enumext_columns_viii_int }
5420     {
5421         \par\noindent
5422         \int_zero:N \__enumext_item_column_pos_viii_int
5423     }
5424 }

```

```

5425         \skip_horizontal:N \l__enumext_columns_sep_viii_dim
5426     }
5427 }

```

(End of definition for \l__enumext_start_item_viii:w and \l__enumext_stop_item_viii:.)

\l__enumext_remove_extra_parsep_viii:

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

```

5428 \cs_new_protected:Nn \l__enumext_remove_extra_parsep_viii:
5429 {
5430     \int_compare:nNnT
5431     {
5432         \int_mod:nn
5433         { \g__enumext_item_count_all_viii_int }
5434         { \l__enumext_columns_viii_int }
5435     }
5436     =
5437     { 0 }
5438     {
5439         \para_end:
5440         \skip_vertical:n { -\l__enumext_itemsep_viii_skip }
5441         \skip_vertical:N \c_zero_skip
5442         \int_gzero:N \g__enumext_item_count_all_viii_int
5443     }
5444 }

```

(End of definition for \l__enumext_remove_extra_parsep_viii:.)

13.48 The command \getkeyans

```

\getkeyans
\__enumext_getkeyans_aux:n
\__enumext_getkeyans:nn

```

The `\getkeyans` command takes a *mandatory argument* of the form $\{\langle store\ name : position \rangle\}$. Retrieve a “single content” stored by `\anskey`, `\anspic*` and `\item*` and `anskey*` from *prop list* defined by `save-ans` key.

```

5445 \NewDocumentCommand \getkeyans { m }
5446 {
5447     \exp_args:Ne \__enumext_getkeyans_aux:n
5448     { \tl_to_str:e { \text_expand:n {#1} } }
5449 }

```

The internal function `__enumext_getkeyans_aux:n` is in charge of *splitting* the *mandatory argument* using “.”. If “.” is omitted it will return an error.

```

5450 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
5451 {
5452     \str_if_in:nnTF {#1} { : }
5453     {
5454         \use:e
5455         {
5456             \cs_set:Npn \exp_not:N \__enumext_tmp:w #1 \c_colon_str ##2 \scan_stop:
5457             { {##1} {##2} }
5458         }
5459         \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
5460     }
5461     { \msg_error:nnn { enumext } { missing-colon } {#1} }
5462 }

```

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*.

```

5463 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
5464 {
5465     \prop_if_exist:cTF { g__enumext_#1_prop }
5466     {
5467         \prop_item:cn { g__enumext_#1_prop } {#2}
5468     }
5469     {
5470         \msg_error:nnn { enumext } { undefined-storage-anskey } {#1}
5471     }
5472 }

```

(End of definition for `\getkeyans`, `__enumext_getkeyans_aux:n`, and `__enumext_getkeyans:nn`. This function is documented on page 19.)

13.49 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⟩]`.

```

5473 \keys_define:nn { enumext / print }
5474 {
5475   print* .code:n = \keys_precompile:neN { enumext / enumext* }
5476               { \__enumext_filter_save_key:n {#1} }
5477               \l__enumext_print_keyans_starred_tl, % starred cmd
5478   print* .initial:n = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5479                       rightmargin=0pt, listparindent=0pt, nosep, label=\arabic*.,
5480                       columns=2, first=\small, font=\small },
5481   print-1 .code:n = \keys_precompile:neN { enumext / level-1 }
5482                  { \__enumext_filter_save_key:n {#1} }
5483                  \l__enumext_print_keyans_i_tl,
5484   print-1 .initial:n = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5485                       rightmargin=0pt, listparindent=0pt, nosep, label=\arabic*.,
5486                       columns=2, first=\small, font=\small },
5487   print-2 .code:n = \keys_precompile:neN { enumext / level-2 }
5488                  { \__enumext_filter_save_key:n {#1} }
5489                  \l__enumext_print_keyans_ii_tl,
5490   print-2 .initial:n = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5491                       rightmargin=0pt, listparindent=0pt, nosep, label=(\alph*),
5492                       first=\small, font=\small },
5493   print-3 .code:n = \keys_precompile:neN { enumext / level-3 }
5494                  { \__enumext_filter_save_key:n {#1} }
5495                  \l__enumext_print_keyans_iii_tl,
5496   print-3 .initial:n = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5497                       rightmargin=0pt, listparindent=0pt, nosep, label=\roman*.,
5498                       first=\small, font=\small },
5499   print-4 .code:n = \keys_precompile:neN { enumext / level-4 }
5500                  { \__enumext_filter_save_key:n {#1} }
5501                  \l__enumext_print_keyans_iv_tl,
5502   print-4 .initial:n = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5503                       rightmargin=0pt, listparindent=0pt, nosep, label=\Alph*.,
5504                       first=\small, font=\small },
5505   print-* .code:n = \keys_precompile:neN { enumext / enumext* }
5506                  { \__enumext_filter_save_key:n {#1} }
5507                  \l__enumext_print_keyans_vii_tl, % starred nested
5508   print-* .initial:n = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5509                       rightmargin=0pt, listparindent=0pt, nosep, label=\arabic*.,
5510                       first=\small, font=\small },
5511 }

```

🔗 The reason for storing *keys* in token lists using `\keys_precompile:neN` is because the keys are set via `\setenumext` but are later executed by running the command `\printkeyans` and they are not handled directly by its *optional argument*, except those related to the *first* opening level.

`\printkeyans`

Create a user command to print “all stored content” in sequence for `\anskey`, `anskey*`, `\item*` and `\anspic*`.

`__enumext_printkeyans:nnn`

Within a group we will run our “precompiled keys” and then call the internal function `__enumext_printkeyans:nnn`.

```

5512 \NewDocumentCommand \printkeyans { s O{ } m }
5513 {
5514   \group_begin:
5515     \bool_set_true:N \l__enumext_print_keyans_cmd_bool
5516     \tl_use:N \l__enumext_print_keyans_i_tl
5517     \tl_use:N \l__enumext_print_keyans_ii_tl
5518     \tl_use:N \l__enumext_print_keyans_iii_tl
5519     \tl_use:N \l__enumext_print_keyans_iv_tl
5520     \tl_use:N \l__enumext_print_keyans_vii_tl
5521     \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
5522     \bool_set_false:N \l__enumext_print_keyans_cmd_bool
5523   \group_end:
5524 }

```

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.

```

5525 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
5526 {
5527   \seq_if_exist:cTF { g__enumext_#3_seq }
5528   {
5529     \seq_if_empty:cF { g__enumext_#3_seq }
5530     {

```

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.

```

5531       \bool_if:nTF {#1}
5532       {
5533         \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
5534         {
5535           \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
5536         }
5537         {
5538           \tl_use:N \l__enumext_print_keyans_starred_tl
5539           \bool_set_true:N \l__enumext_base_line_fix_bool
5540           \bool_set_true:N \l__enumext_print_keyans_star_bool
5541           \begin{enumext*}[#2]
5542             \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5543             \end{enumext*}
5544           \bool_set_false:N \l__enumext_base_line_fix_bool
5545           \bool_set_false:N \l__enumext_print_keyans_star_bool
5546         }
5547       }

```

Otherwise it will open the environment `enumext` passing the *optional argument* to the “*first level*” then map the *sequence*.

```

5548       {
5549         \begin{enumext}[#2]
5550         \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5551         \end{enumext}
5552       }
5553     }
5554   }
5555   {
5556     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
5557   }
5558 }

```

(End of definition for `\printkeyans` and `__enumext_printkeyans:nnn`. This function is documented on page 19.)

13.50 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`.

```

5559 \cs_new:Npn \__enumext_filter_first_level:n #1
5560 {
5561   \use:e
5562   {
5563     \keyval_parse:NNn
5564     \__enumext_filter_first_level_key:n
5565     \__enumext_filter_first_level_pair:nn {#1}
5566   }
5567 }

```

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*`.

```

5568 \cs_new:Npn \__enumext_filter_first_level_key:n #1
5569 {
5570   \str_case:nnF {#1}

```

```

5571     {
5572         { resume } {}
5573         { resume* } {}
5574     }
5575     { , { \exp_not:n {#1} } }
5576 }

```

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.

```

5577 \cs_new:Npn \__enumext_filter_first_level_pair:nn #1#2
5578 {
5579     \str_case:nnF {#1}
5580     {
5581         { series } {}
5582         { resume } {}
5583         { save-ans } {}
5584     }
5585     { , { \exp_not:n {#1} } } = { \exp_not:n {#2} } }
5586 }

```

(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`.

```

5587 \keys_define:nn { enumext / meta-families }
5588 {
5589     enumext-1 .code:n =
5590         {
5591             \keys_set:ne { enumext / level-1 }
5592             {
5593                 \__enumext_filter_first_level:n {#1}
5594             }
5595         } ,
5596     enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
5597     enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
5598     enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
5599     keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} } ,
5600     enumext* .code:n =
5601         {
5602             \keys_set:ne { enumext / enumext* }
5603             {
5604                 \__enumext_filter_first_level:n {#1}
5605             }
5606         } ,
5607     keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
5608     print* .code:n = { \keys_set:nn { enumext / print } { print* = {#1} } } ,
5609     print-1 .code:n = { \keys_set:nn { enumext / print } { print-1 = {#1} } } ,
5610     print-2 .code:n = { \keys_set:nn { enumext / print } { print-2 = {#1} } } ,
5611     print-3 .code:n = { \keys_set:nn { enumext / print } { print-3 = {#1} } } ,
5612     print-4 .code:n = { \keys_set:nn { enumext / print } { print-4 = {#1} } } ,
5613     print-* .code:n = { \keys_set:nn { enumext / print } { print-* = {#1} } } ,
5614     unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
5615 }

```

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

```

5616 \seq_const_from_clist:Nn \c__enumext_all_families_seq
5617 {
5618     enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
5619     keyans*, print-1, print-2, print-3, print-4, print-*, print*,
5620 }

```

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

```

\__enumext_set_parse:n
\__enumext_set_error:nn
5621 \NewDocumentCommand \setenumext { 0{enumext,1} +m }
5622 {
5623     \seq_clear:N \__enumext_setkey_tmpa_seq
5624     \seq_set_from_clist:Nn \__enumext_setkey_tmpb_seq {#1}
5625     \int_set:Nn \__enumext_setkey_tmpa_int
5626     {
5627         \seq_count:N \__enumext_setkey_tmpb_seq
5628     }
5629     \int_compare:nNnTF { \__enumext_setkey_tmpa_int } > { 1 }
5630     {

```

```

5631     \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
5632     \seq_map_function:NN \l__enumext_setkey_tmpb_seq \__enumext_set_parse:n
5633     \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
5634     {
5635         \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
5636     }
5637 }
5638 {
5639     \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
5640 }
5641 \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
5642 { \seq_map_inline:Nn \c__enumext_all_families_seq }
5643 { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
5644 {
5645     \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
5646 }
5647 }

```

Internal functions used by the `\setenumext` command.

```

5648 \cs_new_protected:Npn \__enumext_set_parse:n #1
5649 {
5650     \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
5651     \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
5652     { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
5653     \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
5654     {
5655         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
5656         { \tl_trim_spaces:n {#1} }
5657     }
5658     { \__enumext_set_error:nn {#1} { } }
5659 }
5660 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
5661 { \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.)

13.51 The command `\setenumextmeta`

The command `\setenumextmeta` will be responsible for adding new “meta-keys” for the `enumext` and `enumext*` environments. The implementation code was given by Jonathan P. Spratte (@Skillmon) answer in [Add .meta key to existing keys \(l3keys\)](#).

`\setenumextmeta`

First we will create a prop list `\c__enumext_meta_paths_prop` to handle the *optional argument*.

```

\c__enumext_meta_paths_prop
\__enumext_add_meta_key:nnn
\__enumext_def_meta_key:nnn
\__enumext_def_meta_key:Vnn
5662 \prop_const_from_keyval:Nn \c__enumext_meta_paths_prop
5663 {
5664     {enumext,1} = level-1,
5665     {enumext,2} = level-2,
5666     {enumext,3} = level-3,
5667     {enumext,4} = level-4,
5668     {enumext*} = enumext*
5669 }

```

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

```

5670 \NewDocumentCommand \setenumextmeta { s O{enumext,1} m +m }
5671 {
5672     \str_if_eq:eeTF { \tl_trim_spaces:n {#3} } { unknown }
5673     { \msg_error:nn { enumext } { prohibited-unknown } }
5674     {
5675         \bool_if:nTF {#1}
5676         {
5677             \int_step_inline:nn { 4 }
5678             { \__enumext_add_meta_key:nnn { enumext, ##1 } {#3} {#4} }
5679             \__enumext_add_meta_key:nnn { enumext* } {#3} {#4}
5680         }
5681         { \__enumext_add_meta_key:nnn {#2} {#3} {#4} }
5682     }
5683 }

```

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

```

5684 \cs_new_protected:Npn \__enumext_add_meta_key:nnn #1
5685 {

```



```

5686 \tl_set:Nn \l__enumext_meta_path_tl {#1}
5687 \tl_replace_all:Nnn \l__enumext_meta_path_tl {~} {}
5688 \prop_get:NVNTF
5689 \c__enumext_meta_paths_prop \l__enumext_meta_path_tl \l__enumext_meta_path_tl
5690 { \__enumext_def_meta_key:Vnn \l__enumext_meta_path_tl }
5691 {
5692   \msg_error:nnn { enumext } { unknown-set } {#1}
5693   \use_none:nn
5694 }
5695 }
5696 \cs_new_protected:Npn \__enumext_def_meta_key:nnn #1#2#3
5697 {
5698   \bool_lazy_or:nnTF
5699   { \keys_if_exist_p:nn { enumext / #1 } {#2} }
5700   { \keys_if_exist_p:nn { enumext / enumext* } {#2} }
5701   { \msg_error:nnn { enumext } { already-defined } {#2} }
5702   {
5703     \keys_define:nn { enumext / #1 }
5704     {
5705       #2 .meta:n = {#3},
5706       #2 .value_forbidden:n = true
5707     }
5708   }
5709 }
5710 \cs_generate_variant:Nn \__enumext_def_meta_key:nnn { V }

```

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

13.52 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:nn
\__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.

```

5711 \keys_define:nn { enumext / foreach }
5712 {
5713   before .tl_set:N = \l__enumext_foreach_before_tl,
5714   before .value_required:n = true,
5715   after .tl_set:N = \l__enumext_foreach_after_tl,
5716   after .value_required:n = true,
5717   start .int_set:N = \l__enumext_foreach_start_int,
5718   start .value_required:n = true,
5719   stop .int_set:N = \l__enumext_foreach_stop_int,
5720   stop .value_required:n = true,
5721   step .int_set:N = \l__enumext_foreach_step_int,
5722   step .value_required:n = true,
5723   wrapper .cs_set_protected:Np = \__enumext_foreach_wrapper:n #1,
5724   wrapper .value_required:n = true,
5725   sep .tl_set:N = \l__enumext_foreach_sep_tl,
5726   sep .value_required:n = true,
5727   unknown .code:n = { \__enumext_parse_foreach_keys:n {#1} }
5728 }
5729 \keys_precompile:nnN { enumext / foreach }
5730 {
5731   before={},after={},start=1,step=1,stop=0,wrapper=#1,sep={; }
5732 }
5733 \l__enumext_foreach_default_keys_tl

```

Functions for handling unknown *⟨keys⟩*.

```

5734 \cs_new_protected:Npn \__enumext_parse_foreach_keys:nn #1#2
5735 {
5736   \tl_if_blank:nTF {#2}
5737   {
5738     \msg_error:nnn { enumext } { for-key-unknown } {#1}
5739   }
5740   {
5741     \msg_error:nnnn { enumext } { for-key-value-unknown } {#1} {#2}
5742   }
5743 }
5744 \cs_new_protected:Npn \__enumext_parse_foreach_keys:n #1

```

```

5745 {
5746   \exp_args:NV \__enumext_parse_foreach_keys:nn \l_keys_key_str {#1}
5747 }

```

We create the command.

```

5748 \NewDocumentCommand \foreachkeyans { +0{ } m }
5749 {
5750   \__enumext_foreach_keyans:nn {#1} {#2}
5751 }

```

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

```

5752 \cs_new_protected:Npn \__enumext_foreach_keyans:nn #1 #2
5753 {
5754   \tl_use:N \l__enumext_foreach_default_keys_tl
5755   \keys_set:nn { enumext / foreach } {#1}
5756   \tl_set:Nn \l__enumext_foreach_name_prop_tl {#2}
5757   \prop_if_exist:cF { g__enumext_#2_prop }
5758   {
5759     \msg_error:nnn { enumext } { undefined-storage-anskey } {#2}
5760   }
5761   \int_compare:nNnT { \l__enumext_foreach_stop_int } = { 0 }
5762   {
5763     \int_set:Nn \l__enumext_foreach_stop_int
5764     { \prop_count:c { g__enumext_#2_prop } }
5765   }
5766   \seq_clear:N \l__enumext_foreach_print_seq
5767   \int_step_function:nnnN
5768   { \l__enumext_foreach_start_int }
5769   { \l__enumext_foreach_step_int }
5770   { \l__enumext_foreach_stop_int }
5771   \__enumext_foreach_add_body:n
5772   \seq_use:NV \l__enumext_foreach_print_seq \l__enumext_foreach_sep_tl
5773 }
5774 \cs_new_protected:Npn \__enumext_foreach_add_body:n #1
5775 {
5776   \seq_put_right:Ne \l__enumext_foreach_print_seq
5777   {
5778     \exp_not:V \l__enumext_foreach_before_tl
5779     \__enumext_foreach_wrapper:n
5780     {
5781       \prop_item:cn { g__enumext_ \l__enumext_foreach_name_prop_tl _prop } {#1}
5782     }
5783     \exp_not:V \l__enumext_foreach_after_tl
5784   }
5785 }

```

(End of definition for `\foreachkeyans` and others. This function is documented on page 19.)

13.53 Messages

Message used by package-load for `multicol` and `hyperref` packages.

```

5786 \msg_new:nnn { enumext } { package-load }
5787 {
5788   The~'#1'~package~is~already~loaded.
5789 }
5790 \msg_new:nnn { enumext } { package-not-load }
5791 {
5792   The~'#1'~package~will~be~loaded~as~a~dependency.
5793 }
5794 \msg_new:nnn { enumext } { package-load-foot }
5795 {
5796   The~'#1'~package~is~loaded~with~the~option~'#2'.
5797 }

```

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

```

5798 \msg_new:nnn { enumext } { counters }
5799 {
5800   The~counter~'#1'~is~already~defined~by~some~\\
5801   package~or~macro,~it~cannot~be~continued.
5802 }

```

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

```
5803 \msg_new:nnn { enumext } { unknown-choice }
5804 {
5805   The~value~'#3'~for~'#1'~key~is~invalid~use~('#2').
5806 }
```

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

```
5807 \msg_new:nnnn { enumext } { anskey-env-error }
5808 {
5809   The~environment~'#1'~is~reserved~by ~\
5810   'enumext'~package,~It~is~already~defined.
5811 }
5812 {
5813   The~environment~'#1'~is~defined~internally ~
5814   for~the~'save-ans'~key~with~save-ans~key~active.~See~documentation.\
5815 }
5816 \msg_new:nnn { enumext } { anskey-env-nested }
5817 {
5818   The~'#1'~'#2'~can't~be~nested~\msg_line_context:.
5819 }
```

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

```
5820 \msg_new:nnn { enumext } { store-prop }
5821 {
5822   *~Package~enumext:~Creating ~
5823   \c_backslash_str g__enumext_#1_prop~\msg_line_context:.
5824 }
5825 \msg_new:nnn { enumext } { store-seq }
5826 {
5827   *~Package~enumext:~Creating ~
5828   \c_backslash_str g__enumext_#1_seq~\msg_line_context:.
5829 }
5830 \msg_new:nnn { enumext } { store-int }
5831 {
5832   *~Package~enumext:~Creating ~
5833   \c_backslash_str g__enumext_resume_#1_int~\msg_line_context:.
5834 }
5835 \msg_new:nnn { enumext } { prop-seq-int-hook }
5836 {
5837   *~Package~enumext:~Elements~in ~
5838   \c_backslash_str g__enumext_#1_prop~=#2.\
5839   *~Package~enumext:~Elements~in ~
5840   \c_backslash_str g__enumext_#1_seq~=#3.\
5841   *~Package~enumext:~Value~off ~
5842   \c_backslash_str g__enumext_resume_#1_int~=#4.
5843 }
5844 \msg_new:nnn { enumext } { item-answer-hook }
5845 {
5846   *~Package~enumext:~Value~off ~
5847   \c_backslash_str g__enumext_item_number_int~=#1.\
5848   *~Package~enumext:~Value~off ~
5849   \c_backslash_str g__enumext_item_anskey_int~=#2.\
5850   *~Package~enumext:~Difference~item_number_int~-~item_anskey_int~=#3.
5851 }
```

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

```
5852 \msg_new:nnn { enumext } { invalid-key }
5853 {
5854   The~key~'#1'~is~not~know~the~level~#2.
5855 }
5856 \msg_new:nnn { enumext } { unknown-key-family }
5857 {
5858   Unknown~key~family~`\l_keys_key_str'~for~enumext.
5859 }
```

Messages used in length calculation.

```
5860 \msg_new:nnn { enumext } { width-negative }
5861 {
5862   Ignoring~negative~value~'#1=#2'~\msg_line_context:.
5863   The~key~'#1'~ accepts~values ~>~#opt.
5864 }
5865 \msg_new:nnn { enumext } { width-zero }
5866 {
```

```

5867     Invalid~'#1=#2'~\msg_line_context:.\
5868     The~key~'#1'~ accepts~values ~>~0pt.
5869 }

```

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

```

5870 \msg_new:nnn { enumext } { list-lengths }
5871 {
5872     ****~Lengths~used~by~'#1'~enumext'~level~'#2'~\msg_line_context:~\c_space_tl ****\\
5873     \__enumext_show_length:nnn { dim } { labelsep } {#1}
5874     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
5875     \__enumext_show_length:nnn { dim } { itemindent } {#1}
5876     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
5877     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
5878     \__enumext_show_length:nnn { dim } { listparindent } {#1}
5879     \__enumext_show_length:nnn { skip } { topsep } {#1}
5880     \__enumext_show_length:nnn { skip } { parsep } {#1}
5881     \__enumext_show_length:nnn { skip } { partopsep } {#1}
5882     \__enumext_show_length:nnn { skip } { itemsep } {#1}
5883     *****
5884 }

```

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

```

5885 \msg_new:nnn { enumext } { list-lengths-not-nested }
5886 {
5887     ****~Lengths~used~by~'#2'~environment~\msg_line_context:~\c_space_tl ****\\
5888     \__enumext_show_length:nnn { dim } { labelsep } {#1}
5889     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
5890     \__enumext_show_length:nnn { dim } { itemindent } {#1}
5891     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
5892     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
5893     \__enumext_show_length:nnn { dim } { listparindent } {#1}
5894     \__enumext_show_length:nnn { skip } { topsep } {#1}
5895     \__enumext_show_length:nnn { skip } { parsep } {#1}
5896     \__enumext_show_length:nnn { skip } { partopsep } {#1}
5897     \__enumext_show_length:nnn { skip } { itemsep } {#1}
5898     *****
5899 }

```

Messages used by `ref` key.

```

5900 \msg_new:nnn { enumext } { key-ref-empty }
5901 {
5902     Key~'ref'~need~a~value~in~'#1'~ \msg_line_context:.
5903 }

```

Messages used by `save-ans` key.

```

5904 \msg_new:nnn { enumext } { save-ans-empty }
5905 {
5906     Key~'save-ans'~need~a~value~in~'#1'~ \msg_line_context:.
5907 }
5908 \msg_new:nnn { enumext } { save-ans-log }
5909 {
5910     *~Package~enumext:~Start~#1\c_space_tl with~save-ans=#2~\msg_line_context:.
5911 }
5912 \msg_new:nnn { enumext } { save-ans-log-hook }
5913 {
5914     *~Package~enumext:~Stop~#1\c_space_tl with~save-ans=#2~\msg_line_context:.
5915 }
5916 \msg_new:nnn { enumext } { save-ans-hook }
5917 {
5918     Stop~storing~for~'save-ans=#1'~\msg_line_context:.
5919 }

```

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

```

5920 \msg_new:nnn { enumext } { need-save-ans }
5921 {
5922     Key~'#1'~ works~only~with~the~'save-ans'~key~in~'#2'~ \msg_line_context:.
5923 }
5924 \msg_new:nnn { enumext } { items-same-answer }
5925 {
5926     *****\\
5927     *~Package~enumext:~Checking~answers~in~'#1' ~
5928     for~\c_left_brace_str #2 \c_right_brace_str\\
5929     *~started~#3~and~close~\msg_line_context: : ~

```

```

5930     'OK',~all~items~with~answer.\\
5931     *****
5932 }
5933 \msg_new:nnn { enumext } { item-greater-answer }
5934 {
5935     Checking~answers~in~'#1'~for~\c_left_brace_str #2 \c_right_brace_str\\
5936     started~#3~and~close~\msg_line_context: : ~'NOT~OK'\\
5937     Items~>~Answers.
5938 }
5939 \msg_new:nnn { enumext } { item-less-answer }
5940 {
5941     Checking~answers~in~'#1'~for~\c_left_brace_str #2 \c_right_brace_str\\
5942     started~#3~and~close~\msg_line_context: : ~'NOT~OK'\\
5943     Items~<~Answers.
5944 }

```

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

```

5945 \msg_new:nnn { enumext } { missing-starred }
5946 {
5947     Missing~'\c_backslash_str #1'~#2.
5948 }
5949 \msg_new:nnn { enumext } { many-starred }
5950 {
5951     Many~'\c_backslash_str #1'~#2.
5952 }

```

Messages used by `\printkeyans*` command.

```

5953 \msg_new:nnn { enumext } { print-starred }
5954 {
5955     \c_backslash_str printkeyans*:~ The~sequence~'#1'~already~contains ~
5956     #2~environment~ \msg_line_context:.
5957 }

```

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

```

5958 \msg_new:nnn { enumext } { list-too-deep }
5959 {
5960     Too~deep~nesting ~for~'enumext'~\msg_line_context:~ \\
5961     The~maximum ~level ~of ~nesting ~is~4.
5962 }

```

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

```

5963 \msg_new:nnn { enumext } { anskey-unnumber-item }
5964 {
5965     Can't~store~with~a~unnumbered~\c_backslash_str item~\msg_line_context:.
5966 }
5967 \msg_new:nnn { enumext } { anskey-already-stored }
5968 {
5969     Content~already~stored~for~this~\c_backslash_str item~\msg_line_context:.
5970 }
5971 \msg_new:nnn { enumext } { anskey-empty-arg }
5972 {
5973     Can't~store~empty~content~\msg_line_context:.
5974 }
5975 \msg_new:nnn { enumext } { anskey-wrong-place }
5976 {
5977     Wrong~place~for~command~'\c_backslash_str #1'~\msg_line_context:~ \\
5978     '\c_backslash_str #1'~works~in~the~environment~'#2'.
5979 }
5980 \msg_new:nnn { enumext } { anskey-nested }
5981 {
5982     The~command~\c_backslash_str anskey~ can't~be~nested~\msg_line_context:.
5983 }
5984 \msg_new:nnn { enumext } { anskey-math-mode }
5985 {
5986     #1~can't~work~in~math~mode~\msg_line_context:.
5987 }
5988 \msg_new:nnn { enumext } { anskey-env-wrong }
5989 {
5990     The~environment~anskey*~cannot~use~in~'#1'~\msg_line_context:.
5991 }
5992 \msg_new:nnn { enumext } { anspic-wrong-place }
5993 {
5994     Wrong~place~for~command~'\c_backslash_str #1'~\msg_line_context:~ \\

```

```

5995     '\c_backslash_str #1'~works~in~the~environment~'#2'.
5996 }
5997 \msg_new:nnn { enumext } { command-wrong-place }
5998 {
5999     Wrong~place~for~command~'\c_backslash_str #1'~\msg_line_context:~ \\
6000     '\c_backslash_str #1'~works~outside~the~environment~'#2'.
6001 }
6002 \msg_new:nnnn { enumext } { anskey-env-key-unknown }
6003 {
6004     The~key~'#1'~is~unknown~by~environment~
6005     'anskey* '~and~is~being~ignored.
6006 }
6007 {
6008     The~environment~'anskey* '~does~not~have~a~key~called ~'#1'.\\
6009     Check~that~you~have~spelled~the~key~name~correctly.
6010 }
6011 \msg_new:nnnn { enumext } { anskey-env-key-value-unknown }
6012 {
6013     The~key~'#1=#2'~is~unknown~by~environment ~
6014     'anskey* '~and~is~being~ignored.
6015 }
6016 {
6017     The~environment~'anskey* '~does~not~have~a~key~called ~'#1'.\\
6018     Check~that~you~have~spelled~the~key~name~correctly.
6019 }
6020 \msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
6021 { The~key~'#1'~is~unknown~by~'\c_backslash_str anskey'~and~is~being~ignored. }
6022 {
6023     The~command ~'\c_backslash_str anskey'~does~not~have~a~key~called ~'#1'.\\
6024     Check~that~you~have~spelled~the~key~name~correctly.
6025 }
6026 \msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
6027 { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str anskey'~and~is~being~ignored. }
6028 {
6029     The~command~'\c_backslash_str anskey'~does~not~have~a~key~called ~'#1'.\\
6030     Check~that~you~have~spelled~the~key~name~correctly.
6031 }
6032 \msg_new:nnn { enumext } { overwrite-file }
6033 {
6034     Overwriting~file~'#1'.
6035 }
6036 \msg_new:nnn { enumext } { writing-file }
6037 {
6038     Writing~file~'#1'.
6039 }
6040 \msg_new:nnn { enumext } { not-writing }
6041 {
6042     File~`#1'~already~exists.~Not~writing.
6043 }

```

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

```

6044 \msg_new:nnn { enumext } { keyans-nested }
6045 {
6046     The~environment~'keyans'~can't~be ~nested ~\msg_line_context:.
6047 }
6048 \msg_new:nnn { enumext } { keyans-wrong-level }
6049 {
6050     Wrong~level~position~for~'keyans'~\msg_line_context:~ \\
6051     The~environment~'keyans'~can~only~be~in~the~first~level.
6052 }
6053 \msg_new:nnn { enumext } { wrong-place }
6054 {
6055     Wrong~place~for~'#1'~environment ~\msg_line_context:~ \\
6056     '#1'~is~only~found~with~'#2'~ in ~ 'enumext.
6057 }
6058 \msg_new:nnn { enumext } { keyanspic-nested }
6059 {
6060     The~environment~'keyanspic'~can't~be ~nested~ \msg_line_context:~.
6061 }
6062 \msg_new:nnn { enumext } { keyanspic-wrong-level }
6063 {
6064     Wrong~level~position~for~'keyanspic'~\msg_line_context:~ \\

```

```

6065     The~environment~'keyans'~can~only~be~in~the~first~level.
6066 }
6067 \msg_new:nnn { enumext } { keyanspic-item-cmd }
6068 {
6069     Can't~use ~\c_backslash_str item~in~keyanspic~\msg_line_context:.
6070 }
6071 \msg_new:nnnn { enumext } { keyans-unknown-key }
6072 {
6073     The~key~'#1'~is~unknown~by~environment~
6074     '\l__enumext_envir_name_tl'~and~is~being~ignored.
6075 }
6076 {
6077     The~environment~'\l__enumext_envir_name_tl'~does~not
6078     ~have~a~key~called ~'#1'.\\
6079     Check~that~you~have~spelled~the~key~name~correctly.
6080 }
6081 \msg_new:nnnn { enumext } { keyans-unknown-key-value }
6082 {
6083     The~key~'#1=#2'~is~unknown~by~environment ~
6084     '\l__enumext_envir_name_tl'~and~is~being~ignored.
6085 }
6086 {
6087     The~environment~'\l__enumext_envir_name_tl'~does~not
6088     ~have~a~key~called ~'#1'.\\
6089     Check~that~you~have~spelled~the~key~name~correctly.
6090 }

```

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

```

6091 \msg_new:nnnn { enumext } { starred-unknown-key }
6092 {
6093     The~key~'#1'~is~unknown~by~environment~
6094     '\l__enumext_envir_name_tl'~and~is~being~ignored.
6095 }
6096 {
6097     The~environment~'\l__enumext_envir_name_tl'~does~not
6098     ~have~a~key~called ~'#1'.\\
6099     Check~that~you~have~spelled~the~key~name~correctly.
6100 }
6101 \msg_new:nnnn { enumext } { starred-unknown-key-value }
6102 {
6103     The~key~'#1=#2'~is~unknown~by~environment ~
6104     '\l__enumext_envir_name_tl'~and~is~being~ignored.
6105 }
6106 {
6107     The~environment~'\l__enumext_envir_name_tl'~does~not
6108     ~have~a~key~called ~'#1'.\\
6109     Check~that~you~have~spelled~the~key~name~correctly.
6110 }

```

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

```

6111 \msg_new:nnnn { enumext } { standar-unknown-key }
6112 {
6113     The~key~'#1'~is~unknown~by~environment~'\l__enumext_envir_name_tl' \c_space_tl
6114     ~on~level~\int_use:N \l__enumext_level_int \c_space_tl and~is~being~ignored.
6115 }
6116 {
6117     The~environment~'\l__enumext_envir_name_tl'~does~not
6118     ~have~a~key~called ~'#1'~on~level~\int_use:N \l__enumext_level_int.\\
6119     Check~that~you~have~spelled~the~key~name~correctly.
6120 }
6121 \msg_new:nnnn { enumext } { standar-unknown-key-value }
6122 {
6123     The~key~'#1=#2'~is~unknown~by~environment~'\l__enumext_envir_name_tl' \c_space_tl
6124     ~on~level~\int_use:N \l__enumext_level_int \c_space_tl and~is~being~ignored.
6125 }
6126 {
6127     The~environment~'\l__enumext_envir_name_tl'~does~not
6128     ~have~a~key~called ~'#1'~on~level~\int_use:N \l__enumext_level_int.\\
6129     Check~that~you~have~spelled~the~key~name~correctly.
6130 }

```

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


```

6131 \msg_new:nnnn { enumext } { for-key-unknown }
6132 { The~key~'#1'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored.}
6133 {
6134   The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
6135   Check~that~you~have~spelled~the~key~name~correctly.
6136 }
6137 \msg_new:nnnn { enumext } { for-key-value-unknown }
6138 { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored. }
6139 {
6140   The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
6141   Check~that~you~have~spelled~the~key~name~correctly.
6142 }

```

Messages used by `\getkeyans` command.

```

6143 \msg_new:nnn { enumext } { undefined-storage-anskey }
6144 {
6145   Storage~named~'#1'~is~not~defined~\msg_line_context:.
6146 }

```

Messages used by `\miniright` command.

```

6147 \msg_new:nnn { enumext } { missing-miniright }
6148 {
6149   Missing~'\c_backslash_str miniright'~in~\msg_line_context:.\\
6150   The~key~'mini-env'~need~'\c_backslash_str miniright'.
6151 }
6152 \msg_new:nnn { enumext } { wrong-miniright-place }
6153 {
6154   Wrong~place~for~'\c_backslash_str miniright'~\msg_line_context:.~ \\
6155   Works~in~'enumext'~and~'keyans'~with~key~'mini-env'.
6156 }
6157 \msg_new:nnn { enumext } { wrong-miniright-use }
6158 {
6159   Wrong~use~for~'\c_backslash_str miniright'~\msg_line_context:.~ \\
6160   '\c_backslash_str miniright'~need~a~key~'mini-env'.
6161 }
6162 \msg_new:nnn { enumext } { wrong-miniright-starred }
6163 {
6164   Can't~use ~\c_backslash_str miniright~in~starred~environments~\msg_line_context:.
6165 }
6166 \msg_new:nnn { enumext } { many-miniright-used }
6167 {
6168   Can't~use ~\c_backslash_str miniright~more~than~once~ \msg_line_context:.
6169 }

```

Messages used by `\setenumextmeta` command.

```

6170 \msg_new:nnn { enumext } { unknown-set }
6171 {
6172   Argument~[#1]~is~unknown~by~ \c_backslash_str setenumextmeta~\msg_line_context:.
6173 }
6174 \msg_new:nnn { enumext } { already-defined }
6175 {
6176   The~key~'#1'~is~already~defined~\msg_line_context:.
6177 }
6178 \msg_new:nnn { enumext } { prohibited-unknown }
6179 {
6180   The~name~'unknown'~can't~be~chosen~ for~a~meta~key~\msg_line_context:.
6181 }

```

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

```

6182 \msg_new:nnn { enumext } { nested }
6183 {
6184   The~environment~\l__enumext_envir_name_tl \c_space_tl can't~be~nested~\msg_line_context:.
6185 }
6186 \msg_new:nnn { enumext } { nested-horizontal }
6187 {
6188   The~environment~\l__enumext_envir_name_tl \c_space_tl can't~be~nested~in~'#1'~ \msg_line_cont
6189 }
6190 \msg_new:nnn { enumext } { item-joined }
6191 {
6192   Items~joined~(##1)~>~##2 ~columns ~\msg_line_context:.
6193 }
6194 \msg_new:nnn { enumext } { item-joined-columns }
6195 {

```

```

6196     Not~space~to~join~items~(#1)~>~#2 ~\msg_line_context:.
6197 }

```

Messages used by `resume` key.

```

6198 \msg_new:nnn { enumext } { unknown-series-starred }
6199 {
6200   The~series~'#1'~for~the~resume~key~does~not~exist~in~the~
6201   ~enumext*~environment~ \msg_line_context:.
6202 }
6203 \msg_new:nnn { enumext } { unknown-series-standar }
6204 {
6205   The~series~'#1'~for~the~resume~key~does~not~exist~at~level~\int_use:N \l__enumext_level_int
6206   \c_space_tl of~enumext~environment~ \msg_line_context:.
6207 }
6208 \msg_new:nnnn { enumext-reset } { invalid-clist }
6209 { The~argument~must~have~1~or~2~elements~separated~by~a~comma. }
6210 { Received:~'#1'. }
6211 \msg_new:nnn { enumext-reset } { invalid-single-arg-star }
6212 { The~single~argument~must~be~exactly~'enumext*'*~when~using~a~'*. }
6213 \msg_new:nnnn { enumext-reset } { invalid-single-arg-no-star }
6214 { A~single~argument~is~not~allowed~without~a~'*. }
6215 { Received:~'#1'. }
6216 \msg_new:nnnn { enumext-reset } { out-of-range }
6217 { The~number~must~be~exactly~1,~2,~3~or~4. }
6218 { Received:~'#1'. }
6219 \msg_new:nnnn { enumext-reset } { invalid-package }
6220 { The~first~element~must~be~exactly~'enumext'. }
6221 { Received:~'#1'. }

```

13.54 Finish package

Finish package implementation.

```

6222 \file_input_stop:
6223 </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>	224
<code>\-</code>	224
<code>\\</code> 232, 4569, 4572, 5800, 5809, 5814, 5838, 5840, 5847, 5849, 5862, 5867, 5872, 5887, 5926, 5928, 5930, 5935, 5936, 5941, 5942, 5960, 5977, 5994, 5999, 6008, 6017, 6023, 6029, 6050, 6055, 6064, 6078, 6088, 6098, 6108, 6118, 6128, 6134, 6140, 6149, 6154, 6159	
A	
above	<u>1720</u>
above*	<u>1720</u>
<code>\addvspace</code> 1287, 1315, 1358, 1361, 1529, 1532, 1629, 1635, 1673, 1679, 1700, 1706, 3981, 4153, 4171, 4454, 4458, 4817, 4832, 4878, 4892	
after	<u>1117</u>
align	<u>658</u>
<code>\Alph</code>	44, <u>48</u> , 49
<code>\Alph</code>	600, 728, 772, 832, 5503
<code>\alph</code>	44, <u>48</u> , 49
<code>\alph</code>	601, 726, 5491
<code>\anskey</code>	14, 87, 89, <u>3015</u>
anskey*	15, 3145
<code>\anspic</code>	17, 115, 118, <u>4468</u>
<code>\anspic*</code>	80
<code>\arabic</code>	44
<code>\arabic</code>	599, 725, 771, 5479, 5485, 5509
B	
base-fix	<u>975</u>
<code>\baselineskip</code>	58
<code>\baselineskip</code>	991, 1002
before	<u>1117</u>
before*	<u>1117</u>
beginpenalty	<u>915</u>
below	<u>1720</u>
below*	<u>1720</u>
bool commands:	
<code>\bool_gset_false:N</code> 343, 344, 345, 4834, 4838, 4894	
<code>\bool_gset_true:N</code> 253, 263, 1220, 2448, 2454, 4803, 4835, 4867, 4895	
<code>\bool_if:N</code> TF 393, 403, 420, 494, 501, 510, 517, 531, 544, 1742, 1756, 1769, 1780, 1791, 1802, 1813, 1824, 1838, 1852, 1863, 1901, 1908, 1955, 1999, 2393, 2636, 2646, 2726, 2750, 2757, 2781, 2879, 2901, 2941, 2965, 2969, 3019, 3038, 3062, 3114, 3118, 3148, 3166, 3185, 3201, 3224, 3255, 3270, 3342, 3458, 3492, 3528, 3544, 3565, 3704, 3725, 3771, 3814, 3824, 3858, 3863, 3888, 3897, 3936, 3962, 4012, 4030, 4081, 4136, 4161, 4387, 4452, 4470, 4489, 4540, 4567, 4796, 4812, 4818, 4861, 4875, 4879, 4947, 4968, 4975, 4985, 5073, 5079, 5086, 5102, 5196, 5206, 5320, 5327, 5358, 5374, 5405	
<code>\bool_if:n</code> TF 1680, 1707, 2228, 3514, 3683, 4510, 5531, 5675	
<code>\bool_if_p:N</code> 272, 286, 985, 986, 998, 999, 1652, 2120, 2121, 2136, 2137, 2164, 2165, 2180, 2181, 2406, 2432, 2445, 2446, 2451, 2452, 2814, 2824, 2836, 2851, 2852, 2886, 2927, 2928, 3329, 3330, 3359, 3360, 3372, 3373, 3413, 3414, 3433, 3434, 3717, 3718, 3719, 3909, 3911, 3922, 5349, 5350, 5351	
<code>\bool_lazy_all:n</code> TF 270, 284, 983, 2404, 2430, 2812, 2821, 2834, 2849, 3411, 3431, 3715, 3907, 3920, 5347	
<code>\bool_lazy_and:nn</code> TF 249, 259, 997, 1644, 1651, 2119, 2135, 2163, 2179, 2444, 2450, 2885, 2892, 2926, 3328	
<code>\bool_lazy_or:nn</code> TF 2334, 2341, 3358, 3371, 5698	
<code>\bool_new:N</code> 22, 23, 24, 25, 26, 27, 28, 52, 64, 88, 93, 94, 99, 100, 103, 110, 125, 126, 138, 139, 146, 152, 153, 155, 159, 161, 162, 179, 191, 193	
<code>\bool_not_p:n</code> 250, 260, 987, 1653, 2823, 2887, 2893, 3910, 3923	
<code>\bool_set_eq:NN</code> 3467, 3664, 5026, 5271	
<code>\bool_set_false:N</code> 400, 1009, 2378, 2379, 2411, 2416, 2420, 2424, 2437, 3692, 3877, 4029, 4089, 4176, 4323, 4384, 4530, 4938, 4967, 5023, 5212, 5267, 5268, 5522, 5544, 5545	
<code>\bool_set_true:N</code> 277, 291, 386, 389, 651, 1024, 1726, 1731, 2083, 2096, 2351, 2352, 2668, 2676, 3089, 3461, 3463, 3495, 3497, 3660, 3671, 3685, 3837, 3876, 3916, 3929, 4002, 4086, 4113, 4320, 4512, 4513, 4785, 4850, 4937, 5030, 5037, 5038, 5082, 5210, 5275, 5282, 5283, 5284, 5515, 5539, 5540	
box commands:	
<code>\box_dp:N</code> 1575, 1576, 1579, 1586, 1599, 1607, 1613, 1621, 4398, 4404, 4454, 4551	
<code>\box_ht:N</code> 1358, 1361, 1372, 1373, 1384, 1386, 1401, 1404, 1412, 1413, 1424, 1426, 1441, 1444, 1451, 1452, 1463, 1465, 1480, 1483, 1529, 1532, 1540, 1541, 1549, 1550, 1562, 1564	
<code>\box_ht_plus_dp:N</code> 4393, 4462, 4498	
<code>\box_new:N</code> 61, 148, 149, 186, 192	
<code>\box_use_drop:N</code> 4829, 4890, 5138, 5417	
<code>\box_wd:N</code> 607	
break-col	<u>2985</u> , <u>3071</u>
C	
<code>\c</code>	866, 868, 880, 882
<code>\centering</code>	1682, 1709, 4595, 4822, 4883
check-ans	<u>2370</u>
Document class:	
article	50
clist commands:	
<code>\clist_const:Nn</code>	198
<code>\clist_count:N</code>	2224, 2233, 2238
<code>\clist_item:Nn</code>	2263, 2275, 2276
<code>\clist_map_function:nN</code>	4578
<code>\clist_map_inline:Nn</code> 657, 914, 930, 1116, 1131, 1212, 1736	
<code>\clist_map_inline:nn</code> 36, 45, 57, 69, 77, 90, 102, 141, 170, 197, 635, 688, 708, 1029, 1050, 1226, 1846, 1859, 2256, 2318, 2384, 2563, 2633, 2665, 2809, 3264, 3586, 3601, 3641, 3800, 3803, 3805, 3832, 3844, 3847, 3849, 3868, 5651	
<code>\clist_new:N</code>	47
<code>\clist_set:Nn</code>	2223
<code>\columnbreak</code>	87
<code>\columnbreak</code>	2889
columns	<u>1196</u>
columns-sep	<u>1196</u>

<code>\columnsep</code>	108
<code>\columnsep</code>	3957, 4134
<code>\columnseprule</code>	108
<code>\columnseprule</code>	3960, 4135
Commands provide by enumext :	
<code>\anskey</code> 33, 76, 77, 82–86, 88, 89, 94, 107, 108, 128, 137, 138, 146	
<code>\anspic*</code>	33, 34, 80, 83, 94, 95, 118, 137, 138
<code>\anspic</code>	34, 84, 115, 118, 146
<code>\foreachkeyans</code>	142, 148
<code>\getkeyans</code>	83, 137, 149
<code>\item*</code> 33, 34, 80, 83, 84, 94, 95, 98, 102, 129, 134, 135, 137, 138	
<code>\item</code>	98, 102, 122, 128–130, 133, 134
<code>\miniright</code>	32, 55, 63, 64, 109, 110, 149
<code>\printkeyans*</code>	138
<code>\printkeyans</code>	33, 84, 138
<code>\setenumextmeta</code>	141, 149
<code>\setenumext</code>	33, 138, 140, 141, 144
Counters defined by enumext :	
<code>enumXiii</code>	31, 43
<code>enumXii</code>	31, 43
<code>enumXiv</code>	31, 43
<code>enumXi</code>	31, 43
<code>enumXviii</code>	31, 43
<code>enumXvii</code>	31, 43, 129
<code>enumXvi</code>	31, 43
<code>enumXv</code>	31, 43
<code>\counterwithin</code>	2251
cs commands:	
<code>\cs_generate_variant:Nn</code> . 203, 204, 609, 625, 872, 888, 2718, 2723, 2799, 3144, 3790, 4580, 5710	
<code>\cs_if_exist:NTF</code>	579
<code>\cs_if_exist_p:N</code>	3720, 5352
<code>\cs_new:Nn</code>	218
<code>\cs_new:Npn</code> . 228, 1869, 1878, 1886, 2680, 2689, 2697, 5559, 5568, 5577	
<code>\cs_new_eq:NN</code> . 370, 371, 376, 377, 405, 406, 409, 410	
<code>\cs_new_protected:Nn</code> . 234, 242, 268, 299, 329, 335, 341, 347, 353, 361, 381, 428, 432, 450, 462, 480, 492, 508, 524, 537, 558, 748, 805, 852, 981, 1132, 1136, 1140, 1144, 1148, 1152, 1156, 1160, 1164, 1168, 1172, 1176, 1180, 1184, 1188, 1192, 1227, 1239, 1272, 1289, 1300, 1317, 1343, 1364, 1489, 1515, 1535, 1568, 1590, 1625, 1631, 1737, 1751, 1765, 1776, 1787, 1798, 1809, 1820, 1953, 1971, 1997, 2012, 2108, 2149, 2196, 2319, 2324, 2349, 2389, 2399, 2442, 2457, 2464, 2473, 2478, 2483, 2488, 2497, 2502, 2507, 2724, 2748, 2755, 2779, 2786, 2800, 3036, 3055, 3164, 3183, 3214, 3253, 3268, 3296, 3326, 3354, 3367, 3380, 3409, 3422, 3500, 3510, 3521, 3537, 3553, 3679, 3697, 3731, 3743, 3869, 3905, 3934, 3941, 3971, 3988, 4010, 4035, 4049, 4079, 4103, 4120, 4145, 4159, 4180, 4191, 4359, 4562, 4576, 4581, 4605, 4615, 4646, 4775, 4794, 4840, 4859, 4924, 4954, 4961, 4973, 4983, 5008, 5149, 5194, 5225, 5231, 5252, 5308, 5428	
<code>\cs_new_protected:Npn</code> 205, 206, 210, 214, 413, 577, 594, 604, 610, 729, 773, 837, 859, 873, 1664, 1693, 1895, 1939, 2037, 2058, 2079, 2249, 2254, 2261, 2273, 2329, 2512, 2634, 2644, 2666, 2674, 2710, 2719, 2875, 2938, 2963, 3001, 3005, 3098, 3102, 3135, 3194, 3233, 3306, 3347, 3454, 3473, 3602, 3606, 3620, 3624, 3642, 3646, 3656, 3668, 3713, 3759, 3793, 3835, 3880, 4099, 4368, 4375, 4382, 4487, 4506, 4536, 4677, 4726, 4941, 5014, 5021, 5035, 5043, 5048, 5058, 5218, 5258, 5265, 5280, 5289, 5303, 5345, 5450, 5463, 5525, 5648, 5660, 5684, 5696, 5734, 5744, 5752, 5774	
<code>\cs_new_protected_nopar:Nn</code> . . . 4248, 4292, 4300, 4308, 4993, 5001, 5132, 5237, 5245, 5411	
<code>\cs_new_protected_nopar:Npn</code> . . 4240, 4256, 5064, 5110, 5364, 5385	
<code>\cs_set:Npn</code> 2110, 2151, 2198, 2810, 2847, 5456	
<code>\cs_set_eq:NN</code> . . 3723, 4914, 4915, 5112, 5183, 5184, 5355, 5387	
<code>\cs_set_protected:Nn</code> 1055, 1071, 1084, 1096	
<code>\cs_set_protected:Npn</code> 32, 39, 50, 62, 70, 85, 91, 134, 166, 177, 626, 636, 658, 693, 709, 755, 889, 915, 931, 1011, 1034, 1108, 1117, 1196, 1213, 1720, 1831, 1847, 2310, 2370, 2529, 2564, 2652, 2802, 3257, 3575, 3591, 3634, 3791, 3833	
<code>\cs_to_str:N</code>	596, 619
D	
<code>\d</code>	224
<code>\DeclareDocumentEnvironment</code>	562
dim commands:	
<code>\dim_abs:n</code>	3764, 3769
<code>\dim_add:Nn</code>	3406, 4402, 4640, 4671
<code>\dim_compare:nNnTF</code> . . 1057, 1073, 1086, 1098, 1376, 1388, 1416, 1428, 1455, 1467, 1544, 1552, 1666, 1695, 2943, 2951, 3401, 3761, 3766, 3772, 3778, 3780, 3782, 3946, 3993, 4107, 4124, 4377, 4617, 4633, 4648, 4664, 4777, 4842, 5316	
<code>\dim_compare:nTF</code>	2911, 4038, 4183
<code>\dim_eval:n</code>	991, 4460, 4547
<code>\dim_gset_eq:NN</code>	4786, 4851
<code>\dim_gzero:N</code>	4837, 4897
<code>\dim_new:N</code> . 58, 65, 66, 67, 87, 130, 131, 143, 150, 151, 185, 187, 188, 194	
<code>\dim_set:Nn</code> . 607, 1025, 2945, 2953, 3388, 3392, 3397, 3403, 3490, 3764, 3769, 3771, 3774, 3775, 3779, 3781, 3784, 3785, 3787, 3949, 3996, 4037, 4109, 4126, 4182, 4391, 4496, 4583, 4619, 4626, 4650, 4657, 4712, 4761, 4779, 4844, 5060, 5318	
<code>\dim_set_eq:NN</code> 716, 762, 829, 3485, 3802, 3846, 3957, 4134, 4719, 4722, 4723, 4768, 4771, 4772, 5053, 5124, 5399	
<code>\dim_sub:Nn</code>	4043, 4188, 4635, 4666
<code>\dim_use:N</code> . 1058, 1066, 1667, 1677, 2789, 2792, 2797, 2955, 3505, 3507, 3560, 3947, 3951, 3952, 3954, 3994, 3999, 4000, 4006, 4040, 4045	
<code>\dim_zero:N</code>	3838, 3960, 4135, 4405
<code>\dim_zero_new:N</code>	576
<code>\c_zero_dim</code> 1060, 1074, 1087, 1099, 1667, 1695, 2913, 2943, 2951, 3388, 3401, 3761, 3766, 3772, 3779, 3947, 3994, 4040, 4107, 4124, 4185, 4377, 4617, 4633, 4648, 4664, 4777, 4842, 5316	
<code>\dimeval</code>	2598
E	
<code>\end</code> . . . 2752, 2783, 3978, 4150, 4442, 4597, 5533, 5543, 5551	
end internal commands:	
<code>\end__enumext_mini_page</code> . 1675, 1702, 4021, 4170, 4801, 4865, 4891	
<code>\endlist</code>	371
<code>\endminipage</code>	377
<code>endpenalty</code>	915
<code>enumext</code>	5, 4056
enumext internal commands:	
<code>__enumext_add_meta_key:nnn</code> . . 141, 5662, 5678, 5679, 5681, 5684	

```

\__enumext_add_pre_parsep: . 56, 1237, 1239, 1239
\__enumext_after_args_exec: 54, 1132, 1144, 4070
\__enumext_after_args_exec_v: 1148, 1160, 4208
\__enumext_after_args_exec_vii: . . 1164, 1188
\__enumext_after_args_exec_viii: . . . . . 1192
\__enumext_after_env:nn 93, 111, 124, 132, 210, 210,
    550, 554, 4075, 4810, 4873, 5165
\__enumext_after_hyperref: . . . 39, 379, 379, 381
\l__enumext_after_list_args_v_tl . . . . . 1162
\l__enumext_after_list_args_vii_tl 1190, 5130
\l__enumext_after_list_args_viii_tl . . 1194,
    5409
\__enumext_after_list_vii: 124, 127, 4922, 4961,
    4961
\__enumext_after_list_viii: . . . 133, 5192, 5231,
    5231
\__enumext_after_stop_list: 54, 110, 1132, 1140,
    4026
\__enumext_after_stop_list_v: 1148, 1156, 4177
\l__enumext_after_stop_list_v_tl . . . . . 1158
\__enumext_after_stop_list_vii: . . 127, 1164,
    1180, 4964
\l__enumext_after_stop_list_vii_tl . . . 1182
\__enumext_after_stop_list_viii: . 1184, 5234
\l__enumext_after_stop_list_viii_tl . . . 1186
\l__enumext_align_label_pos_v_str 3384, 3749
\l__enumext_align_label_pos_X_str . . . . . 70
\l__enumext_align_label_vii_str . . . . . 5099
\l__enumext_align_label_viii_str . . . . . 5378
\l__enumext_align_label_X_str . . . . . 177
\c__enumext_all_envs_clist . . 198, 657, 914, 930,
    1116, 1131, 1212, 1736
\c__enumext_all_families_seq . . 140, 5616, 5642
\__enumext_anskey_env_file_if_writable:n 91,
    3112, 3112
\__enumext_anskey_env_file_if_-
    writable:nTF . . . . . 3112, 3137
\__enumext_anskey_env_file_write:nn 91, 3135,
    3144, 3199
\l__enumext_anskey_env_force_eol_bool . . 93,
    3085, 3201
\c__enumext_anskey_env_hidden_space_str 33,
    93, 113, 3205
\l__enumext_anskey_env_overwrite_bool 3093,
    3118
\__enumext_anskey_env_safe_inner: . 92, 3159,
    3164, 3183
\__enumext_anskey_env_safe_inner:n . . . . 92
\__enumext_anskey_env_safe_outer: . 92, 3147,
    3164, 3164
\__enumext_anskey_env_unknown:n 91, 3096, 3098,
    3098
\__enumext_anskey_env_unknown:nn . 3098, 3100,
    3102
\l__enumext_anskey_level_int . . 16, 3057, 3058
\__enumext_anskey_safe_inner: . 90, 3030, 3036,
    3055
\__enumext_anskey_safe_inner:n . . . . . 89
\__enumext_anskey_safe_outer: . 89, 3017, 3036,
    3036
\__enumext_anskey_show_wrap_arg:n . 88, 2938,
    2938, 2967, 2982
\__enumext_anskey_show_wrap_left:n 88, 2883,
    2963, 2963
\__enumext_anskey_unknown:n 89, 2985, 2999, 3001
\__enumext_anskey_unknown:nn . 2985, 3003, 3005
\__enumext_anskey_wrapper:n . . . . . 2595, 2961
\l__enumext_anspic_above_int . 142, 4584, 4585,
    4587
\__enumext_anspic_args:nnn 118, 120, 4468, 4484,
    4562
\l__enumext_anspic_args_seq 118, 120, 142, 4482,
    4596, 4609
\l__enumext_anspic_below_int . 142, 4584, 4585,
    4588
\l__enumext_anspic_body_box . . . 142, 4495, 4498
\__enumext_anspic_body_dim:n . . 119, 4468, 4487,
    4539
\l__enumext_anspic_body_htdp_dim . . 119, 142,
    4496, 4550
__enumext_anspic_exec: . . . . . 4468
\__enumext_anspic_exec: . . . 118, 121, 4437, 4605
\__enumext_anspic_label:nn 119, 4468, 4506, 4542,
    4557
\l__enumext_anspic_label_above_bool . . . 142,
    4320, 4323, 4387, 4452, 4489, 4540, 4567
\l__enumext_anspic_label_box . . 142, 4390, 4393
\l__enumext_anspic_label_htdp_dim . 117, 142,
    4391, 4397, 4462, 4549
\__enumext_anspic_label_pos:nnn . . 120, 4468,
    4536, 4565
\l__enumext_anspic_label_sep_skip 4330, 4399,
    4463, 4552, 4569
\l__enumext_anspic_layout_style_tl 4332, 4607,
    4612
\l__enumext_anspic_mini_pos_str . . 142, 4321,
    4324, 4594
\l__enumext_anspic_mini_width_dim 142, 4508,
    4583, 4594
\__enumext_anspic_print:n 120, 4468, 4576, 4580,
    4609, 4612
\__enumext_anspic_row:n . . 120, 4468, 4578, 4581
\__enumext_anspic_start_list_tag: 4264, 4292,
    4564
\__enumext_anspic_stop_list_tag: . 4264, 4308,
    4574
\__enumext_anspic_stop_start_list_tag: 4264,
    4300, 4566
\__enumext_at_begin_document:n . . 39, 206, 206,
    368, 374
\l__enumext_base_line_fix_bool 51, 139, 977, 986,
    1009, 5539, 5544
\__enumext_before_args_exec: 54, 109, 127, 1132,
    1132, 3991
\__enumext_before_args_exec_v: 1148, 1148, 4106
\__enumext_before_args_exec_vii: . 1164, 1164,
    4958
\__enumext_before_args_exec_viii: 1168, 5228
\__enumext_before_env:nn . . . . . 210, 214
\__enumext_before_keys_exec: . . 54, 1132, 1136,
    4066
\__enumext_before_keys_exec_v: 1148, 1152, 4204
\__enumext_before_keys_exec_vii . . . . . 1164
\__enumext_before_keys_exec_vii: . 1172, 4908
\__enumext_before_keys_exec_viii: 1176, 5177
\__enumext_before_list: . . 109, 3988, 3988, 4060
\__enumext_before_list_v: . . . 4103, 4103, 4199
\__enumext_before_list_vii: . . . 127, 4903, 4954,
    4954

```

```

\__enumext_before_list_viii: .. 133, 5173, 5225,
    5225
\l__enumext_before_no_starred_key_v_tl 1154
\l__enumext_before_no_starred_key_vii_-
    tl ..... 1174
\l__enumext_before_no_starred_key_viii_-
    tl ..... 1178
\l__enumext_before_starred_key_v_tl ... 1150
\l__enumext_before_starred_key_vii_tl . 1166
\l__enumext_before_starred_key_viii_tl 1170
\__enumext_calc_hspace:NNNNNN 105, 3759, 3759,
    3790, 3795, 3839
\__enumext_check_ans_active: 77, 109, 127, 2389,
    2389, 3992, 4957
\g__enumext_check_ans_item_tl ..... 95
\g__enumext_check_ans_key_bool 78, 79, 152, 343,
    2448, 2454, 3224
\l__enumext_check_ans_key_bool 78, 2374, 2379,
    2445, 2451
\__enumext_check_ans_key_hook: .. 78, 110, 127,
    2442, 2442, 4027, 4965
\__enumext_check_ans_level: . 77, 78, 2389, 2395,
    2399
\__enumext_check_ans_log: 79, 93, 2488, 2488, 3228
\__enumext_check_ans_log_msg_greater: 2488,
    2494, 2507
\__enumext_check_ans_log_msg_less: 2488, 2492,
    2497
\__enumext_check_ans_log_msg_same_ok: 2488,
    2493, 2502
\__enumext_check_ans_msg_greater: 2464, 2470,
    2483
\__enumext_check_ans_msg_less: 2464, 2468, 2473
\__enumext_check_ans_msg_same_ok: 2464, 2469,
    2478
\__enumext_check_ans_show: .. 79, 93, 2464, 2464,
    3226
\l__enumext_check_answers_bool 76, 77, 89, 92, 98,
    152, 2352, 2378, 2393, 2726, 2750, 2757, 2781, 3019,
    3148, 3342, 3458, 3492, 5079
\__enumext_check_starred_cmd:n 37, 80, 95, 132,
    2512, 2512, 4211, 4450, 5191
\g__enumext_check_starred_cmd_int . 102, 152,
    2515, 2521, 2526, 3677, 4518, 5315
\l__enumext_check_start_line_env_tl . 37, 152,
    306, 314, 322, 2518, 2524, 2527
\l__enumext_columns_sep_v_dim 4124, 4126, 4134
\l__enumext_columns_sep_vii_dim .. 4617, 4619,
    4628, 4640, 4716, 5146
\l__enumext_columns_sep_viii_dim . 4648, 4650,
    4659, 4671, 4765, 5425
\l__enumext_columns_v_int 1509, 1527, 1698, 4122,
    4130, 4142, 4147
\l__enumext_columns_vii_int .. 4622, 4625, 4629,
    4638, 4680, 4684, 4687, 4693, 4699, 4703, 5140, 5154
\l__enumext_columns_viii_int . 4653, 4656, 4660,
    4669, 4729, 4733, 4736, 4742, 4748, 4752, 5419, 5434
\l__enumext_counter_i_tl ..... 32, 586
\l__enumext_counter_ii_tl ..... 32, 587
\l__enumext_counter_iii_tl ..... 32, 588
\l__enumext_counter_iv_tl ..... 32, 589
\g__enumext_counter_styles_tl . 31, 44, 58, 597,
    615
\l__enumext_counter_v_tl ..... 32, 590
\l__enumext_counter_vi_tl ..... 32, 591
\l__enumext_counter_vii_tl ..... 32, 592
\l__enumext_counter_viii_tl ..... 32, 593
\l__enumext_current_widest_dim 31, 58, 621, 717,
    763, 830
\__enumext_def_meta_key:nnn .. 141, 5662, 5690,
    5696, 5710
\__enumext_default_item:n ... 3454, 3454, 3518
\__enumext_define_counter:Nn . 31, 577, 577, 586,
    587, 588, 589, 590, 591, 592, 593
\__enumext_endminipage: . 39, 368, 377, 571, 4831,
    5134, 5413
\g__enumext_envir_name_tl 37, 22, 278, 292, 351,
    2322, 2327, 2337, 2476, 2481, 2486, 2500, 2505, 2510
\l__enumext_envir_name_tl . 36, 37, 101, 22, 248,
    258, 305, 313, 321, 3596, 4355, 6074, 6077, 6084, 6087,
    6094, 6097, 6104, 6107, 6113, 6117, 6123, 6127, 6184,
    6188
\__enumext_execute_after_env: .. 38, 75, 79, 93,
    3214, 3214, 4077, 5167
\__enumext_fake_item_indent: . 1055, 1055, 3823
\l__enumext_fake_item_indent_v_dim 1074, 1079
\l__enumext_fake_item_indent_v_tl 1076, 3661,
    3665, 3672
\__enumext_fake_item_indent_vii: . 1055, 1084,
    3857
\l__enumext_fake_item_indent_vii_dim . 1087,
    1091
\l__enumext_fake_item_indent_vii_tl .. 1089,
    5129
\__enumext_fake_item_indent_viii: 1055, 1096,
    3862
\l__enumext_fake_item_indent_viii_dim 1099,
    1103
\l__enumext_fake_item_indent_viii_tl . 1101,
    5404
\l__enumext_fake_item_indent_X_tl ..... 91
\__enumext_fake_make_label_vii:n . 129, 5064,
    5064, 5121
\__enumext_fake_make_label_viii:n 5345, 5364,
    5396
\__enumext_filter_first_level:n .. 139, 5559,
    5559, 5593, 5604
\__enumext_filter_first_level_key:n 139, 5559,
    5564, 5568
\__enumext_filter_first_level_pair:nn . 140,
    5559, 5565, 5577
\__enumext_filter_save_key:n .. 83, 2641, 2649,
    2672, 2678, 2680, 2680, 5476, 5482, 5488, 5494, 5500,
    5506
\__enumext_filter_save_key_key:n .. 83, 2680,
    2685, 2689
\__enumext_filter_save_key_pair:nn 83, 2680,
    2686, 2697
\__enumext_filter_series:n 68, 1869, 1869, 1920,
    1931, 1945, 1950
\__enumext_filter_series_key:n 68, 1869, 1874,
    1878
\__enumext_filter_series_pair:nn .. 68, 1869,
    1875, 1886
\__enumext_first_item_tmp_vii: 126, 128, 4914,
    4993, 4993
\__enumext_first_item_tmp_viii: .. 133, 5183,
    5237, 5237
\g__enumext_footnote_standar_arg_seq .. 171,

```


[445](#), [456](#), [459](#)
[\g__enumext_footnote_standar_int](#) [171](#), [439](#), [442](#),
[444](#), [447](#)
[\g__enumext_footnote_standar_int_seq](#) .. [171](#),
[447](#), [452](#), [455](#), [460](#)
[\g__enumext_footnote_starred_arg_seq](#) .. [171](#),
[475](#), [486](#), [489](#)
[\g__enumext_footnote_starred_int](#) [171](#), [469](#), [472](#),
[474](#), [477](#)
[\g__enumext_footnote_starred_int_seq](#) .. [171](#),
[477](#), [482](#), [485](#), [490](#)
[__enumext_footnotes_key_bool](#) [39](#)
[\l__enumext_footnotes_key_bool](#) [34](#), [39](#), [161](#), [389](#),
[393](#), [400](#), [501](#), [517](#), [531](#), [544](#)
[__enumext_footnotetext:nn](#) .. [428](#), [428](#), [457](#), [487](#)
[__enumext_foreach_add_body:n](#) [143](#), [5711](#), [5771](#),
[5774](#)
[\l__enumext_foreach_after_tl](#) [5715](#), [5783](#)
[\l__enumext_foreach_before_tl](#) [5713](#), [5778](#)
[\g__enumext_foreach_default_keys_tl](#) ... [142](#)
[\l__enumext_foreach_default_keys_tl](#) ... [120](#),
[5733](#), [5754](#)
[__enumext_foreach_keyans:nn](#) . [143](#), [5711](#), [5750](#),
[5752](#)
[\l__enumext_foreach_name_prop_tl](#) . [120](#), [5756](#),
[5781](#)
[\l__enumext_foreach_print_seq](#) [120](#), [5766](#), [5772](#),
[5776](#)
[\l__enumext_foreach_sep_tl](#) [5725](#), [5772](#)
[\l__enumext_foreach_start_int](#) [5717](#), [5768](#)
[\l__enumext_foreach_step_int](#) [5721](#), [5769](#)
[\l__enumext_foreach_stop_int](#) . [5719](#), [5761](#), [5763](#),
[5770](#)
[__enumext_foreach_wrapper:n](#) [5723](#), [5779](#)
[__enumext_getkeyans:nn](#) .. [137](#), [5445](#), [5459](#), [5463](#)
[__enumext_getkeyans_aux:n](#) [137](#), [5445](#), [5447](#), [5450](#)
[\l__enumext_hyperref_bool](#) . [34](#), [39](#), [40](#), [161](#), [386](#),
[403](#), [420](#), [2928](#), [3330](#), [5073](#)
[__enumext_hypertarget:nn](#) [40](#), [379](#), [405](#), [409](#), [425](#)
[__enumext_if_is_int:n](#) [222](#)
[__enumext_if_is_int:nTF](#) [222](#), [861](#), [875](#)
[__enumext_internal_mini_page:](#) [42](#), [107](#), [126](#), [558](#),
[558](#), [3872](#), [4927](#)
[__enumext_is_not_nested:](#) . [31](#), [36](#), [107](#), [126](#), [242](#),
[242](#), [3871](#), [4926](#)
[__enumext_is_on_first_level:](#) . [31](#), [37](#), [107](#), [126](#),
[242](#), [268](#), [3878](#), [4939](#)
[\g__enumext_item_anskey_int](#) [89](#), [95](#), [152](#), [338](#), [365](#),
[366](#), [2461](#), [2877](#), [3344](#)
[__enumext_item_answer_diff:](#) [79](#), [93](#), [2457](#), [2457](#),
[3221](#)
[\g__enumext_item_answer_diff_int](#) [79](#), [152](#), [339](#),
[2459](#), [2466](#), [2490](#)
[\l__enumext_item_column_pos_vii_int](#) [128](#), [4687](#),
[4693](#), [4699](#), [4703](#), [4710](#), [5004](#), [5140](#), [5143](#)
[\l__enumext_item_column_pos_viii_int](#) .. [133](#),
[4736](#), [4742](#), [4748](#), [4752](#), [4759](#), [5248](#), [5419](#), [5422](#)
[\l__enumext_item_column_pos_X_int](#) [177](#)
[\g__enumext_item_count_all_vii_int](#) [128](#), [4711](#),
[5005](#), [5154](#), [5162](#)
[\g__enumext_item_count_all_viii_int](#) [133](#), [4760](#),
[5249](#), [5433](#), [5442](#)
[\g__enumext_item_count_all_X_int](#) [177](#)
[\g__enumext_item_number_bool](#) [152](#)
[\l__enumext_item_number_bool](#) [78](#), [159](#), [2411](#), [2416](#),
[2420](#), [2424](#), [2437](#), [3062](#), [3185](#), [3461](#), [3495](#), [5082](#)
[\g__enumext_item_number_int](#) .. [78](#), [152](#), [337](#), [364](#),
[366](#), [2410](#), [2415](#), [2419](#), [2423](#), [2436](#), [2461](#), [3460](#), [3494](#),
[5081](#)
[__enumext_item_peek_args_vii:](#) [128](#), [5001](#), [5006](#),
[5008](#)
[__enumext_item_peek_args_viii:](#) [133](#), [134](#), [5245](#),
[5250](#), [5252](#)
[__enumext_item_starred_exec:](#) . [99](#), [3473](#), [3500](#),
[3542](#), [3563](#)
[__enumext_item_starred_exec:nn](#) .. [3473](#), [3473](#),
[3516](#)
[\l__enumext_item_starred_vii_bool](#) [5023](#), [5037](#),
[5086](#)
[\l__enumext_item_starred_viii_bool](#) [5267](#), [5282](#),
[5374](#), [5405](#)
[\l__enumext_item_starred_X_bool](#) [177](#)
[__enumext_item_std:w](#) . [39](#), [98](#), [102](#), [368](#), [372](#), [3464](#),
[3470](#), [3498](#), [3661](#), [3665](#), [3672](#)
[\g__enumext_item_symbol_aux_tl](#) . [98](#), [124](#), [3478](#),
[3481](#), [3506](#), [3550](#), [3570](#)
[\g__enumext_item_symbol_aux_vii_tl](#) [5045](#), [5088](#),
[5091](#), [5095](#), [5097](#)
[\g__enumext_item_symbol_aux_X_tl](#) [177](#)
[\l__enumext_item_symbol_sep_vii_dim](#) .. [5053](#),
[5060](#), [5094](#), [5096](#)
[\l__enumext_item_symbol_vii_tl](#) [5091](#)
[\l__enumext_item_text_vii_box](#) [5113](#), [5138](#)
[\l__enumext_item_text_viii_box](#) ... [5388](#), [5417](#)
[\l__enumext_item_text_X_box](#) [177](#)
[\l__enumext_item_width_vii_dim](#) ... [4626](#), [4635](#),
[4714](#), [4722](#), [4723](#)
[\l__enumext_item_width_viii_dim](#) .. [4657](#), [4666](#),
[4763](#), [4771](#), [4772](#)
[\l__enumext_item_width_X_dim](#) [177](#)
[\l__enumext_item_wrap_key_bool](#) [103](#), [152](#), [3414](#),
[3434](#), [3685](#), [3692](#), [3719](#), [4512](#), [4530](#), [5268](#), [5283](#), [5351](#)
[\l__enumext_itemindent_X_dim](#) [62](#)
[\l__enumext_itemsep_i_skip](#) ... [1370](#), [1377](#), [1380](#),
[1382](#), [1389](#), [1393](#), [1396](#), [1398](#), [1538](#), [1545](#), [1547](#), [1548](#),
[1553](#), [1557](#), [1559](#), [1560](#)
[\l__enumext_itemsep_ii_skip](#) .. [1410](#), [1417](#), [1420](#),
[1422](#), [1429](#), [1433](#), [1436](#), [1438](#)
[\l__enumext_itemsep_iii_skip](#) . [1449](#), [1456](#), [1459](#),
[1461](#), [1468](#), [1472](#), [1475](#), [1477](#)
[\l__enumext_itemsep_vii_skip](#) [5160](#)
[\l__enumext_itemsep_viii_skip](#) [5440](#)
[\l__enumext_joined_item_aux_vii_int](#) .. [4708](#),
[4709](#), [4710](#), [4711](#), [4717](#)
[\l__enumext_joined_item_aux_viii_int](#) . [4757](#),
[4758](#), [4759](#), [4760](#), [4766](#)
[\l__enumext_joined_item_aux_X_int](#) [177](#)
[__enumext_joined_item_vii:w](#) .. [128](#), [5001](#), [5011](#),
[5012](#), [5014](#)
[\l__enumext_joined_item_vii_int](#) .. [4679](#), [4680](#),
[4683](#), [4685](#), [4691](#), [4696](#), [4701](#), [4706](#), [4708](#), [4714](#)
[__enumext_joined_item_viii:w](#) [134](#), [5245](#), [5255](#),
[5256](#), [5258](#)
[\l__enumext_joined_item_viii_int](#) . [4728](#), [4729](#),
[4732](#), [4734](#), [4740](#), [4745](#), [4750](#), [4755](#), [4757](#), [4763](#)
[\l__enumext_joined_item_X_int](#) [177](#)
[\l__enumext_joined_width_vii_dim](#) . [4712](#), [4719](#),
[4722](#), [5115](#), [5123](#)

`\l__enumext_joined_width_viii_dim` 4761, 4768, 4771, 5390, 5398
`\l__enumext_joined_width_X_dim` 177
`__enumext_keyans_addto_prop:n` 94, 3233, 3233, 3674, 4515
`__enumext_keyans_addto_seq:n` . 95, 3306, 3306, 3676, 4517
`__enumext_keyans_addto_seq_link:` 3306, 3324, 3326, 5314
`__enumext_keyans_default_item:n` . 102, 3656, 3656, 3693
`\l__enumext_keyans_env_bool` 22, 3910, 3923, 4086, 4176
`__enumext_keyans_fake_item_indent:` .. 1055, 1071, 3813
`\l__enumext_keyans_level_h_int` .. 132, 16, 790, 814, 3046, 3174, 3284, 4933, 5200, 5201
`\l__enumext_keyans_level_int` .. 16, 1658, 3042, 3170, 3279, 3424, 4085, 4090, 4478
`__enumext_keyans_make_label:` . 103, 3697, 3697, 3811
`__enumext_keyans_make_label_box:` 3697, 3701, 3706, 3743
`__enumext_keyans_make_label_std:` 3697, 3709, 3731
`__enumext_keyans_mini_right_cmd:n` 64, 1660, 1693, 1693
`__enumext_keyans_mini_set_vskip:` 61
`__enumext_keyans_minipage_add_space:` 1489, 1515, 4115
`__enumext_keyans_minipage_set_skip:` . 1489, 1489, 1517
`__enumext_keyans_multi_addvspace:` 1289, 1300, 4139
`__enumext_keyans_multi_set_vskip:` 57, 1289, 1289, 1302
`__enumext_keyans_multicols_start:` 4103, 4118, 4120
`__enumext_keyans_multicols_stop:` 1697, 4103, 4145, 4174
`__enumext_keyans_name_and_start:` 31, 37, 132, 299, 299, 4087, 4366, 5205
`__enumext_keyans_parse_keys:n` 4099, 4099, 4198
`__enumext_keyans_pic_arg_two:` 117, 4359, 4382, 4413
`\l__enumext_keyans_pic_level_int` .. 16, 1639, 3050, 3178, 3236, 3274, 3309, 4361, 4362
`__enumext_keyans_pic_parse_keys:n` 4359, 4368, 4412
`__enumext_keyans_pic_safe_exec:` . 116, 4359, 4359, 4411
`__enumext_keyans_pic_skip_abs:N` . 117, 4359, 4375, 4386
`__enumext_keyans_pos_mark_set:` 96, 3380, 3380, 3417, 3449
`__enumext_keyans_pre_itemsep_skip:` .. 1489, 1508, 1535
`__enumext_keyans_redefine_item:` . 103, 3679, 3679, 3810
`__enumext_keyans_ref:` 48, 837, 852, 3812
`__enumext_keyans_ref:n` 48, 834, 837, 837
`__enumext_keyans_safe_exec:` . 4079, 4079, 4197
`__enumext_keyans_save_item_opt:n` .. 96, 102, 3347, 3347, 3670, 4514
`__enumext_keyans_set_item_width:` 113, 4180, 4180, 4207
`__enumext_keyans_show_ans:` 97, 3380, 3409, 3736, 3751, 4519
`__enumext_keyans_show_item_opt:` 96, 102, 3347, 3354, 3673, 4527
`__enumext_keyans_show_item_opt_viii:` .. 96, 3347, 3367, 5407
`__enumext_keyans_show_pos:` 97, 3380, 3422, 3737, 3752, 4520
`__enumext_keyans_starred_item:n` . 102, 3668, 3668, 3688
`__enumext_keyans_starred_item_star:` .. 135, 5280, 5308, 5376
`__enumext_keyans_start_counter:` . 4191, 4191, 4206
`__enumext_keyans_store_ref:` .. 94, 3253, 3253, 3675, 4516, 5312
`__enumext_keyans_store_ref_aux_i:` 94, 3253, 3265, 3268
`__enumext_keyans_store_ref_aux_ii:` 95, 3253, 3294, 3296
`__enumext_keyans_unknown_keys:n` . 3591, 3597, 3602, 4356
`__enumext_keyans_unknown_keys:nn` 3591, 3604, 3606
`__enumext_keyans_wraper_label:n` 103
`__enumext_keyans_wraper_label_viii:n` 5345, 5345, 5381
`__enumext_keyans_wrapper_item_v:n` 3720, 3723
`__enumext_keyans_wrapper_item_viii:n` 5352, 5356
`__enumext_keyans_wrapper_label:n` 3697, 3713, 3739, 3754, 4524
`__enumext_keyans_wrapper_opt_v:n` 3362
`__enumext_keyans_wrapper_opt_viii:n` .. 3375
`\l__enumext_label_copy_i_tl` .. 2843, 3272, 3277, 3282, 3287
`\l__enumext_label_copy_v_tl` 3282
`\l__enumext_label_copy_vi_tl` 3277
`\l__enumext_label_copy_vii_tl` 2819, 2830, 2859, 3272
`\l__enumext_label_copy_viii_tl` 3287
`\l__enumext_label_copy_X_tl` 163
`\l__enumext_label_fill_left_v_tl` 3735
`\l__enumext_label_fill_left_X_tl` 91
`\l__enumext_label_fill_right_v_tl` 3740
`\l__enumext_label_fill_right_X_tl` 91
`\l__enumext_label_font_style_v_tl` 3738, 3753, 4523, 4531
`\l__enumext_label_font_style_vii_tl` ... 5101
`\l__enumext_label_font_style_viii_tl` .. 5380
`\l__enumext_label_i_tl` 709
`\l__enumext_label_ii_tl` 709
`\l__enumext_label_iii_tl` 709
`\l__enumext_label_iv_tl` 709
`__enumext_label_style:Nnn` 31, 44, 610, 610, 625, 714, 760, 825, 827
`\l__enumext_label_v_tl` 95, 822, 3241, 3314, 3383, 4201, 4390
`\l__enumext_label_vi_tl` 95, 822, 3238, 3311, 4524, 4532
`\l__enumext_label_vii_tl` . 755, 5032, 5055, 5062
`\l__enumext_label_viii_tl` 755, 5277, 5306, 5310

```

\l__enumext_label_width_by_box .. 58, 606, 607
\__enumext_label_width_by_box:Nn 44, 604, 604,
    609, 621, 885, 3382
\l__enumext_labelsep_v_dim ... 3403, 4129, 4402,
    4526
\l__enumext_labelsep_vii_dim . 2945, 4621, 4631,
    4715, 4997, 5053, 5108, 5117
\l__enumext_labelsep_viii_dim 4652, 4662, 4764,
    5241, 5318, 5383, 5392
\l__enumext_labelwidth_v_dim . 830, 3393, 3398,
    3419, 3451, 3749, 4129, 4402, 4521
\l__enumext_labelwidth_vii_dim ... 2948, 4621,
    4630, 4715, 4997, 5099, 5116
\l__enumext_labelwidth_viii_dim .. 4652, 4661,
    4764, 5241, 5325, 5342, 5378, 5391
\l__enumext_leftmargin_tmp_v_bool . 117, 4384
\l__enumext_leftmargin_tmp_X_bool ..... 62
\l__enumext_leftmargin_tmp_X_dim ..... 62
\l__enumext_leftmargin_X_dim ..... 62
\__enumext_level: 218, 218, 739, 741, 750, 752, 1058,
    1062, 1066, 1134, 1138, 1142, 1146, 1229, 1231, 1233,
    1235, 1277, 1279, 1281, 1283, 1287, 1321, 1327, 1332,
    1334, 1337, 1340, 1353, 1356, 1667, 1671, 1677, 1740,
    1742, 1744, 1747, 1754, 1756, 1758, 1761, 1901, 1917,
    1919, 1921, 1923, 1943, 1945, 1976, 1985, 1990, 1993,
    2046, 2050, 2083, 2636, 2638, 2640, 2668, 2669, 2671,
    2728, 2736, 2740, 2744, 2955, 2959, 3463, 3464, 3468,
    3469, 3470, 3478, 3486, 3487, 3490, 3497, 3498, 3502,
    3505, 3507, 3541, 3543, 3544, 3546, 3549, 3560, 3561,
    3564, 3565, 3567, 3916, 3929, 3936, 3944, 3947, 3949,
    3951, 3952, 3953, 3954, 3957, 3962, 3968, 3974, 3981,
    3994, 3996, 3999, 4000, 4002, 4006, 4012, 4040, 4045,
    4051, 4053, 4063, 4065
\l__enumext_level_h_int 126, 16, 251, 274, 287, 776,
    807, 1646, 1906, 1926, 1947, 2060, 2094, 2133, 2177,
    2212, 2407, 2427, 2838, 3924, 4928, 4929
\l__enumext_level_int . 107, 16, 220, 261, 273, 288,
    560, 1241, 1366, 1645, 1899, 1915, 1941, 1958, 2039,
    2049, 2081, 2117, 2127, 2130, 2161, 2171, 2174, 2204,
    2208, 2210, 2401, 2433, 2815, 2825, 2831, 2837, 2844,
    2853, 2858, 3216, 3827, 3873, 3874, 3885, 3896, 3914,
    3927, 3958, 4094, 4474, 4977, 4987, 5213, 6114, 6118,
    6124, 6128, 6205
\__enumext_list_arg_two_i: ..... 3791
\__enumext_list_arg_two_ii: ..... 3791
\__enumext_list_arg_two_iii: ..... 3791
\__enumext_list_arg_two_iv: ..... 3791
\__enumext_list_arg_two_v: 103, 3791, 4203, 4385
\__enumext_list_arg_two_vii: ..... 3833, 4907
\__enumext_list_arg_two_viii: ..... 3833, 5176
\l__enumext_listoffset_v_dim . 4131, 4185, 4188
\l__enumext_listparindent_vii_dim 5124, 5128
\l__enumext_listparindent_viii_dim 5399, 5403
\__enumext_log_answer_vars: . 38, 353, 361, 3223
\__enumext_log_global_vars: . 38, 353, 353, 3222
\__enumext_make_label: .... 99, 3521, 3521, 3821
\__enumext_make_label_box: ... 3521, 3525, 3530,
    3553
\__enumext_make_label_std: ... 3521, 3533, 3537
\l__enumext_mark_answer_sym_tl 85, 2580, 2794,
    2971, 3405, 5322, 5329
\l__enumext_mark_answer_sym_v_tl . 3405, 3437
\l__enumext_mark_answer_sym_viii_tl ... 5322
\l__enumext_mark_position_str 124, 2586, 2587,
    2588, 2792, 3407, 5323, 5340
\l__enumext_mark_position_v_str .. 124, 3407
\l__enumext_mark_position_viii_str 124, 5323,
    5340
\l__enumext_mark_ref_sym_tl .. 2568, 2933, 3338
\l__enumext_mark_sep_tmpa_dim 124, 3383, 3393,
    3398
\l__enumext_mark_sep_tmpb_dim 124, 3388, 3392,
    3397, 3406
\l__enumext_mark_sym_sep_dim . 2583, 2943, 2945,
    2948, 2951, 2953
\l__enumext_mark_sym_sep_v_dim ... 3401, 3403,
    3406, 3419, 3451
\l__enumext_mark_sym_sep_viii_dim 5316, 5318,
    5325, 5342
\l__enumext_meta_path_tl . 120, 5686, 5687, 5689,
    5690
\c__enumext_meta_paths_prop ..... 141, 5662
\__enumext_mini_addvspace_vii: 63, 1625, 1625,
    4789
\__enumext_mini_addvspace_viii: 63, 1625, 1631,
    4854
__enumext_mini_env* ..... 558
\__enumext_mini_page 1677, 1704, 4006, 4116, 4791,
    4856, 4877
\__enumext_mini_right_cmd:n 64, 1662, 1664, 1664
\__enumext_mini_set_vskip_vii: 62, 1568, 1568,
    1627
\__enumext_mini_set_vskip_viii: 62, 1568, 1590,
    1633
\__enumext_minipage:w 39, 368, 376, 565, 4814, 5123,
    5398
\l__enumext_minipage_active_v_bool 4113, 4136,
    4161
\g__enumext_minipage_active_vii_bool .. 124,
    4803, 4812, 4834
\l__enumext_minipage_active_vii_bool . 4785,
    4796
\g__enumext_minipage_active_viii_bool 4867,
    4875, 4894
\l__enumext_minipage_active_viii_bool 4850,
    4861
\g__enumext_minipage_active_X_bool ... 177
\l__enumext_minipage_active_X_bool .... 78
\__enumext_minipage_add_space: . 59, 109, 1317,
    1343, 4004
\g__enumext_minipage_after_skip 78, 1572, 1584,
    4832, 4892
\l__enumext_minipage_after_skip .. 58, 110, 78,
    1330, 1370, 1372, 1377, 1380, 1384, 1389, 1393, 1396,
    1400, 1412, 1417, 1420, 1424, 1429, 1433, 1436, 1440,
    1451, 1456, 1459, 1463, 1468, 1472, 1475, 1479, 1491,
    1505, 1538, 1540, 1545, 1547, 1549, 1553, 1557, 1559,
    1561, 1592, 1605, 1619, 1673, 1700, 4171
\g__enumext_minipage_center_vii_bool . 4818,
    4835
\g__enumext_minipage_center_viii_bool 4879,
    4895
\g__enumext_minipage_center_X_bool ... 177
\l__enumext_minipage_hsep_v_dim ..... 4111
\l__enumext_minipage_hsep_vii_dim .... 4783
\l__enumext_minipage_hsep_viii_dim ... 4848
\l__enumext_minipage_left_skip 78, 1492, 1570,
    1575, 1579, 1593, 1597, 1611, 1629, 1635
\l__enumext_minipage_left_v_dim .. 4109, 4116

```

`\l__enumext_minipage_left_vii_dim` 4779, 4791
`\l__enumext_minipage_left_viii_dim` 4844, 4856
`\l__enumext_minipage_left_X_dim` 78
`\g__enumext_minipage_right_skip` 78, 1571, 1576, 1580, 4817, 4878
`\l__enumext_minipage_right_skip` . 58, 78, 1319, 1325, 1330, 1332, 1334, 1493, 1494, 1500, 1505, 1506, 1507, 1512, 1594, 1601, 1615, 1679, 1706
`\l__enumext_minipage_right_v_dim` . 1695, 1704, 4107, 4111
`\g__enumext_minipage_right_vii_dim` 124, 4787, 4814, 4837
`\l__enumext_minipage_right_vii_dim` 124, 4777, 4782, 4788
`\g__enumext_minipage_right_viii_dim` . 4852, 4877, 4897
`\l__enumext_minipage_right_viii_dim` . 4842, 4847, 4853
`\g__enumext_minipage_right_X_dim` 177
`\g__enumext_minipage_right_X_skip` 177
`__enumext_minipage_set_skip:` . 58, 1317, 1317, 1345
`\g__enumext_minipage_stat_int` . . 109, 78, 1684, 1711, 4003, 4014, 4019, 4114, 4163, 4168
`\l__enumext_minipage_temp_skip` 78, 1391, 1401, 1404, 1431, 1441, 1444, 1470, 1480, 1483, 1555, 1562, 1564
`\l__enumext_miniright_code_vii_box` 4825, 4829
`\g__enumext_miniright_code_vii_tl` 124, 4820, 4827, 4836
`\l__enumext_miniright_code_viii_box` . 4886, 4890
`\g__enumext_miniright_code_viii_tl` 4881, 4888, 4896
`\l__enumext_miniright_code_X_box` 177
`\l__enumext_mode_box_bool` 630, 3528, 3704
`__enumext_multi_addvspace:` 57, 108, 1272, 1272, 3965
`__enumext_multi_set_vskip:` 56, 1227, 1227, 1274
`\l__enumext_multicols_above_ii_skip` . . 1246
`\l__enumext_multicols_above_iii_skip` . . 1255
`\l__enumext_multicols_above_iv_skip` . . 1264
`\l__enumext_multicols_above_v_skip` 1291, 1305, 1315, 1506
`\l__enumext_multicols_above_X_skip` 70
`\l__enumext_multicols_below_ii_skip` . . 1373, 1382, 1386, 1398, 1403
`\l__enumext_multicols_below_iii_skip` . 1413, 1422, 1426, 1438, 1443
`\l__enumext_multicols_below_iv_skip` . . 1452, 1461, 1465, 1477, 1482
`\l__enumext_multicols_below_v_skip` 1295, 1309, 1507, 1541, 1548, 1550, 1560, 1563, 4153
`\l__enumext_multicols_below_X_skip` 70
`\g__enumext_multicols_right_X_skip` 70
`__enumext_multicols_start:` 108, 109, 3941, 3941, 4008
`__enumext_multicols_stop:` 109, 1669, 3971, 3971, 4024
`__enumext_nested_base_line_fix:` 51, 107, 975, 981, 3892
`__enumext_newlabel:nn` 34, 40, 86, 413, 413, 2869, 3300
`\l__enumext_newlabel_arg_one_tl` 34, 40, 86, 94, 163, 2862, 2870, 2932, 3289, 3301, 3336
`\l__enumext_newlabel_arg_two_tl` 34, 40, 85, 163, 2818, 2828, 2841, 2856, 2871, 3276, 3281, 3286, 3302
`\g__enumext_not_key_series_vii_tl` 1949, 1950, 2214, 2217
`\g__enumext_not_key_series_X_tl` 46
`__enumext_parse_foreach_keys:n` . . 5711, 5727, 5744
`__enumext_parse_foreach_keys:nn` . 5711, 5734, 5746
`__enumext_parse_keys:n` 51, 68, 3880, 3880, 4059
`__enumext_parse_keys_vii:n` 68, 4902, 4941, 4941
`__enumext_parse_keys_viii:n` . 5172, 5218, 5218
`__enumext_parse_save_key:n` 83, 2661, 2666, 2666
`__enumext_parse_save_key_vii:n` 83, 2656, 2666, 2674
`__enumext_parse_series:n` . . 68, 107, 127, 1895, 1895, 3890, 3899, 4949
`__enumext_parse_store_keys:n` 107
`\l__enumext_parsep_i_skip` 1244, 1248
`\l__enumext_parsep_ii_skip` 1253, 1257
`\l__enumext_parsep_iii_skip` 1262, 1266
`\l__enumext_parsep_vii_skip` 5125
`\l__enumext_parsep_viii_skip` 5400
`\l__enumext_partopsep_v_skip` . 1307, 1311, 1502, 1525
`\l__enumext_partopsep_viii_skip` 1603
`__enumext_phantomsection:` 40, 379, 406, 410, 426
`__enumext_pre_itemsep_skip:` 58, 59, 1335, 1364, 1364
`__enumext_print_footnote:` . . 428, 450, 514, 519
`__enumext_print_footnote_mini:` 428, 480, 541, 546
`__enumext_print_footnote_standar:` 492, 508, 572
`__enumext_print_footnote_starred:` 492, 537, 552, 556
`__enumext_print_keyans_box:NN` 85, 2786, 2786, 2799, 2947, 2958, 3418, 3450, 5324, 5341
`\l__enumext_print_keyans_cmd_bool` 124, 1838, 1852, 1863, 3888, 3897, 4030, 4947, 4968, 5515, 5522
`\l__enumext_print_keyans_i_tl` . . . 5483, 5516
`\l__enumext_print_keyans_ii_tl` . . 5489, 5517
`\l__enumext_print_keyans_iii_tl` . . 5495, 5518
`\l__enumext_print_keyans_iv_tl` . . 5501, 5519
`\l__enumext_print_keyans_star_bool` . 51, 139, 124, 987, 999, 5540, 5545
`\l__enumext_print_keyans_starred_tl` 138, 139, 124, 5477, 5538
`\l__enumext_print_keyans_vii_tl` 138, 5507, 5520
`\l__enumext_print_keyans_X_tl` 124
`__enumext_printkeyans:nnn` 138, 139, 5512, 5521, 5525
`__enumext_redefine_item:` . 99, 3510, 3510, 3820
`\l__enumext_ref_key_arg_t` 46
`\l__enumext_ref_key_arg_tl` 37, 731, 732, 744, 775, 778, 786, 792, 800, 839, 840, 848
`\l__enumext_ref_the_count_tl` . 46, 37, 737, 743, 783, 786, 797, 800, 845, 848
`__enumext_register_default_label_wd:Nn` 594, 594, 599, 600, 601, 602, 603
`__enumext_remove_extra_parsep_vii:` . . 4921, 5149, 5149
`__enumext_remove_extra_parsep_viii:` . 5190, 5428, 5428
`\l__enumext_renew_counter_v_tl` . 846, 854, 856

```

\l__enumext_renew_counter_vii_tl 784, 809, 811
\l__enumext_renew_counter_viii_tl . 798, 816,
    818
\l__enumext_renew_counter_X_tl . . . . . 37
\__enumext_renew_footnote: . . 428, 432, 498, 503
\__enumext_renew_footnote_mini: 428, 462, 528,
    533
\__enumext_renew_footnote_standar: 492, 492,
    564
\__enumext_renew_footnote_starred: 492, 524,
    5119, 5394
\l__enumext_reset_args_clist . . 46, 2223, 2224,
    2233, 2238, 2263, 2275, 2276
\__enumext_reset_count_resume:nn . 2249, 2258,
    2266, 2283, 2288, 2293, 2298
\__enumext_reset_count_resume_all:n . 2230,
    2254
\__enumext_reset_global_bool: . . 329, 332, 341
\__enumext_reset_global_int: . . . 329, 331, 335
\__enumext_reset_global_tl: . . . 329, 333, 347
\__enumext_reset_global_vars: . 38, 93, 329, 329,
    3230
\l__enumext_reset_level_int 49, 2276, 2279, 2284,
    2289, 2294, 2299, 2303
\l__enumext_reset_name_str 46, 2263, 2264, 2275,
    2277, 2307
\__enumext_resume_counter: 72, 2079, 2088, 2101,
    2108
\__enumext_resume_counter:n . 68, 71, 2043, 2048,
    2064, 2069, 2079, 2079, 2202, 2216
\__enumext_resume_counter_series: 72, 73, 2079,
    2091, 2104, 2149
\__enumext_resume_last:n 68, 69, 1895, 1903, 1910,
    1939
\__enumext_resume_save_counter: . . . 110, 127
\__enumext_resume_series:n . 71, 1854, 2037, 2037
\__enumext_resume_series_vii:n 71, 1865, 2037,
    2058
\__enumext_resume_star: . . . 73, 1840, 2196, 2196
\l__enumext_resume_vii_bool . . . . . 1908, 2096
\g__enumext_resume_vii_int . . . . . 2025, 2145
\l__enumext_resume_X_bool . . . . . 46
\g__enumext_resume_X_int . . . . . 46
\l__enumext_rightmargin_vii_dim . . 4633, 4637,
    4642
\l__enumext_rightmargin_viii_dim . 4664, 4668,
    4673
\__enumext_safe_exec: . . 42, 107, 3869, 3869, 4058
\__enumext_safe_exec_vii: . 42, 4901, 4924, 4924
\__enumext_safe_exec_viii: 132, 5171, 5194, 5194
\__enumext_scan_tokens:n . . . 93, 205, 205, 3211
\__enumext_second_part: . . 110, 4010, 4010, 4073
\__enumext_second_part_v: . . . 4103, 4159, 4212
\l__enumext_series_name_str . 69, 107, 127, 1835,
    1897, 1917, 1919, 1921, 1923, 1928, 1930, 1932, 1934,
    1973, 1976, 1980, 2014, 2017, 2021, 3884, 4945
\l__enumext_series_name_tl . 69, 71, 72, 46, 1978,
    1990, 1993, 2019, 2030, 2033, 2084, 2085, 2086, 2097,
    2098, 2099, 2153, 2157, 2191
\g__enumext_series_name_X_tl . . . . . 46
\__enumext_set_error:nn . . . . . 5621, 5658, 5660
\__enumext_set_item_width: 110, 4035, 4035, 4069
\__enumext_set_parse:n . . . . . 5621, 5632, 5648
\l__enumext_setkey_tmpa_int . . . 115, 5625, 5629
\l__enumext_setkey_tmpa_seq . . . 115, 5623, 5633,
    5639, 5641, 5643, 5655
\l__enumext_setkey_tmpa_tl . . . . 115, 5631, 5635
\l__enumext_setkey_tmpb_seq . . . 115, 5624, 5627,
    5631, 5632
\l__enumext_setkey_tmpb_tl 115, 5650, 5652, 5653
\l__enumext_show_answer_bool . 2555, 2574, 2965,
    3359, 3372, 3413, 3718, 5320, 5350
\__enumext_show_length:nnn . . 53, 228, 228, 5873,
    5874, 5875, 5876, 5877, 5878, 5879, 5880, 5881, 5882,
    5888, 5889, 5890, 5891, 5892, 5893, 5894, 5895, 5896,
    5897
\l__enumext_show_pos_tmp_int . 124, 3426, 3429,
    3444
\l__enumext_show_position_bool . . . 2558, 2577,
    2969, 3360, 3373, 3433, 5327
\g__enumext_standar_bool 36, 107, 22, 250, 253, 272,
    344, 494, 510, 1955, 2432, 2446, 2823, 2836, 2851,
    3911
\l__enumext_standar_bool 107, 110, 22, 1653, 2824,
    3876, 4029, 4938
\l__enumext_standar_first_bool 37, 107, 22, 277,
    2120, 2164, 2335, 2342
\__enumext_standar_item_vii:w . 128, 129, 5001,
    5019, 5021
\__enumext_standar_item_viii:w 134, 5245, 5263,
    5265
\__enumext_standar_ref: . . . . 46, 729, 748, 3822
\__enumext_standar_ref:n . . . . . 721, 729, 729
\__enumext_standar_reset:n . . . . . 2240, 2273
\__enumext_standar_save_counter: . . 69, 1953,
    1953, 4032
\__enumext_standar_save_counter_aux: . 1953,
    1957, 1968, 1971
\__enumext_standar_unknown_keys:n 3634, 3638,
    3642
\__enumext_standar_unknown_keys:nn 3634, 3644,
    3646
\__enumext_standard_ref:n . . . . . 46
\g__enumext_starred_bool 36, 126, 22, 260, 263, 286,
    345, 1652, 1999, 2406, 2452, 2814, 3270, 4838
\l__enumext_starred_bool 126, 127, 132, 22, 2852,
    2887, 2893, 2941, 3877, 4937, 4967, 5206, 5210
\__enumext_starred_columns_set_vii: . . 4615,
    4615, 4912
\__enumext_starred_columns_set_viii: . 4615,
    4646, 5181
\l__enumext_starred_first_bool 37, 126, 22, 291,
    985, 998, 2136, 2180, 2335, 2342
\__enumext_starred_item_vii:w . 128, 129, 5001,
    5018, 5035
\__enumext_starred_item_vii_aux_i:w . . 5001,
    5040, 5043
\__enumext_starred_item_vii_aux_ii:w . 5001,
    5041, 5046, 5048
\__enumext_starred_item_vii_aux_iii:w 5001,
    5051, 5058
\__enumext_starred_item_viii:w 134, 5262, 5280,
    5280
\__enumext_starred_item_viii_aux_i:w . . 134,
    5280, 5286, 5289
\__enumext_starred_item_viii_aux_ii:w . 134,
    5280, 5287, 5301, 5303
\__enumext_starred_joined_item_vii:n 122, 128,
    4677, 4677, 5016

```

```

\__enumext_starred_joined_item_viii:n . 122,
    134, 4677, 4726, 5260
\__enumext_starred_ref: . . . . 47, 773, 805, 3854
\__enumext_starred_ref:n . . . . 47, 767, 773, 773
\__enumext_starred_reset:n . . . . . 2235, 2261
\__enumext_starred_save_counter: . . 69, 1953,
    1997, 4970
\__enumext_starred_save_counter_aux: . 1953,
    2001, 2009, 2012
\__enumext_starred_unknown_keys:n 3616, 3618,
    3620
\__enumext_starred_unknown_keys:nn 3616, 3622,
    3624
\__enumext_start_counter: . . . 4049, 4049, 4068
\__enumext_start_from:NNn 48, 859, 859, 872, 894,
    900
\l__enumext_start_i_int . . . . . 2123, 2167
\__enumext_start_item_tmp_vii: 126, 4915, 5001,
    5001
\__enumext_start_item_tmp_viii: . . 5184, 5245,
    5245
\__enumext_start_item_vii:w 129, 130, 5027, 5032,
    5055, 5062, 5110, 5110
\__enumext_start_item_viii:w . 134, 5272, 5277,
    5306, 5385, 5385
\g__enumext_start_line_tl 37, 22, 279, 293, 350,
    2476, 2481, 2486, 2500, 2505, 2510
\__enumext_start_list:nn 39, 104, 368, 370, 4062,
    4200, 4905, 5174
\__enumext_start_list_tag:n . . 4214, 4240, 5120,
    5395
\__enumext_start_mini_vii: 127, 4775, 4775, 4959
\__enumext_start_mini_viii: . . . 133, 4840, 4840,
    5229
\__enumext_start_save_ans_msg: . . 75, 76, 2319,
    2319, 2344
\__enumext_start_store_level: . 107, 3905, 3905,
    4061
\__enumext_start_store_level_vii: 128, 4904,
    4973, 4973
\l__enumext_start_vii_int 2139, 2145, 2183, 2189
\l__enumext_start_X_int . . . . . 91
\__enumext_stop_item_tmp_vii: . . 126, 128, 130,
    4914, 4920, 5003, 5112
\__enumext_stop_item_tmp_viii: 133, 5183, 5189,
    5247, 5387
\__enumext_stop_item_vii: 130, 131, 5110, 5112,
    5132
\__enumext_stop_item_viii: . . . 5385, 5387, 5411
\__enumext_stop_list: 39, 124, 127, 368, 371, 3976,
    3984, 4149, 4156, 4798, 4806, 4863, 4870
\__enumext_stop_list_tag:n . . . 4214, 4256, 5135,
    5414
\__enumext_stop_mini_vii: 124, 127, 4775, 4794,
    4963
\__enumext_stop_mini_viii: 133, 4840, 4859, 5233
\__enumext_stop_save_ans_msg: . 75, 2319, 2324,
    3220
\__enumext_stop_start_list_tag: . . 4214, 4248,
    5122, 5397
\__enumext_stop_store_level: . . 108, 109, 3934,
    3934, 3977, 3985
\__enumext_stop_store_level_vii: 124, 127, 128,
    4799, 4807, 4973, 4983
\l__enumext_store_active_bool 33, 76, 103, 2121,
    2137, 2165, 2181, 2351, 3038, 3166, 3909, 3922, 4081,
    4089, 4470, 4975, 4985, 5196, 5212
\__enumext_store_active_keys:n . 82, 107, 2634,
    2634, 3902
\__enumext_store_active_keys_vii:n . 82, 127,
    2634, 2644, 4951
\__enumext_store_addto_prop:n 83, 94, 2710, 2710,
    2718, 2878, 3251, 5311
\__enumext_store_addto_seq:n 84, 95, 2719, 2719,
    2723, 2730, 2744, 2752, 2761, 2775, 2783, 2936, 3341
\__enumext_store_anskey_arg:n . . 87, 89, 92, 93,
    2875, 2875, 3031, 3209
\l__enumext_store_anskey_arg_tl . . 33, 87, 108,
    2884, 2889, 2891, 2896, 2903, 2906, 2916, 2921, 2924,
    2930, 2936
\__enumext_store_anskey_env:n . 93, 3160, 3164,
    3194
\l__enumext_store_anskey_env_tl . . 33, 93, 108,
    3196, 3198, 3200, 3203, 3211
\__enumext_store_anskey_safe_outer: . . 90, 92
\l__enumext_store_columns_break_bool . 2886,
    2987, 3073
\l__enumext_store_current_label_tl 33, 94, 95,
    134, 103, 3235, 3238, 3241, 3247, 3249, 3251, 3308,
    3311, 3314, 3320, 3322, 3332, 3341, 5291, 5296, 5297,
    5310, 5311, 5313
\l__enumext_store_current_opt_arg_tl . 33, 96,
    134, 103, 3351, 3356, 3363, 3369, 3376, 5299
\__enumext_store_internal_ref: . . 85, 87, 2800,
    2800, 2881
\l__enumext_store_item_join_int . . 2894, 2898,
    2990, 3076
\l__enumext_store_item_star_bool . 2901, 2992,
    3078
\l__enumext_store_item_symbol_sep_dim 2913,
    2918, 2997, 3083
\l__enumext_store_item_symbol_tl . 2904, 2908,
    2995, 3081
\l__enumext_store_keyans_item_opt_sep_v_-
    tl . . . . . 3245, 3247, 3318, 3320
\l__enumext_store_keyans_item_opt_sep_-
    viii_tl . . . . . 5294, 5296
\__enumext_store_level_close: . 84, 2724, 2748,
    3938
\__enumext_store_level_close_vii: . 84, 2755,
    2779, 4989
\__enumext_store_level_open: 84, 108, 2724, 2724,
    3917, 3930
\__enumext_store_level_open_vii: . . 84, 2755,
    2755, 4979
\g__enumext_store_name_tl 33, 76, 103, 349, 356,
    357, 358, 359, 2327, 2353, 2475, 2480, 2485, 2499,
    2504, 2509, 3218
\l__enumext_store_name_tl 33, 76, 77, 103, 1960,
    1963, 1982, 2002, 2005, 2023, 2125, 2141, 2169, 2185,
    2322, 2331, 2332, 2353, 2354, 2356, 2357, 2359, 2361,
    2362, 2364, 2366, 2367, 2391, 2712, 2714, 2721, 2864,
    2865, 2977, 3291, 3292, 3443, 5335
\l__enumext_store_ref_key_bool 87, 2571, 2879,
    2927, 3255, 3329
\l__enumext_store_save_key_vii_bool . . 2646,
    2676
\l__enumext_store_save_key_vii_tl 2648, 2649,
    2677, 2678, 2759, 2767, 2771, 2775

```


<code>\l__enumext_store_save_key_X_bool</code> .. 82, 124	5235
<code>\l__enumext_store_save_key_X_tl</code> 82, 124	<code>\l__enumext_vspace_below_viii_skip</code> 1822, 1826, 1828
<code>\l__enumext_store_upper_level_X_bool</code> .. 124	<code>__enumext_widest_from:nNNn</code> .. 49, 873 , 873, 888, 907
<code>__enumext_storing_exec:</code> .. 76, 2329 , 2345 , 2349	<code>\g__enumext_widest_label_tl</code> 31, 44, 58 , 614, 618, 622
<code>__enumext_storing_set:n</code> 75, 76, 2314 , 2329 , 2329	<code>\l__enumext_wrap_label_opt_v_bool</code> 3664
<code>\l__enumext_the_counter_v_tl</code> 845	<code>\l__enumext_wrap_label_opt_vii_bool</code> 129, 5026
<code>\l__enumext_the_counter_vii_tl</code> 783	<code>\l__enumext_wrap_label_opt_viii_bool</code> .. 134 , 5271
<code>\l__enumext_the_counter_viii_tl</code> 797	<code>\l__enumext_wrap_label_opt_X_bool</code> 91
<code>\l__enumext_the_counter_X_tl</code> 37	<code>\l__enumext_wrap_label_v_bool</code> 3660, 3664 , 3671 , 3717 , 3725 , 4513
<code>__enumext_tmp:n</code> 32, 36, 39, 45, 50, 57, 62, 69, 70, 77, 85, 90, 91, 102, 134, 141, 166, 170, 177, 197, 626, 635, 1831, 1846, 1847, 1859, 2110, 2127, 2130, 2151, 2171, 2174, 2198, 2210, 2310, 2318, 2370, 2388, 2564, 2633, 2652, 2665, 2802, 2809, 2810, 2831, 2844, 2847, 2858, 3257, 3264, 3591, 3601, 3634, 3641, 3791, 3832, 3833, 3868	<code>\l__enumext_wrap_label_vii_bool</code> .. 129, 5026 , 5030 , 5038 , 5102
<code>__enumext_tmp:nn</code> 636, 657, 658, 692, 693, 708, 889, 914, 915, 930, 1011, 1033, 1034, 1054, 1108, 1116, 1117, 1131, 1196, 1212, 1213, 1226, 1720, 1736, 2529, 2563, 3575, 3590	<code>\l__enumext_wrap_label_viii_bool</code> . 134 , 5271 , 5275 , 5284 , 5349 , 5358
<code>__enumext_tmp:nnn</code> 709, 725, 726, 727, 728, 755, 771, 772	<code>\l__enumext_wrap_label_X_bool</code> 91
<code>__enumext_tmp:nnnnnn</code> 931, 956, 959, 962, 964, 966, 969 , 972	<code>__enumext_wrapper_label_v:n</code> . 3723, 3727, 4532
<code>__enumext_tmp:w</code> 5456, 5459	<code>__enumext_wrapper_label_vii:n</code> 5104
<code>\l__enumext_tmpa_vii_int</code> 4625, 4628, 4637, 4668	<code>__enumext_wrapper_label_viii:n</code> .. 5356 , 5360
<code>\l__enumext_tmpa_viii_int</code> 4656, 4659	<code>\l__enumext_write_anskey_env_bool</code> .. 33, 108 , 3089 , 3114
<code>\l__enumext_tmpa_X_dim</code> 177	<code>\l__enumext_write_anskey_env_file_iow</code> .. 33, 108 , 3139 , 3140 , 3141
<code>\l__enumext_tmpa_X_int</code> 177	<code>\l__enumext_write_anskey_env_file_name_tl</code> 33, 108 , 3090, 3200
<code>\l__enumext_topsep_v_skip</code> 1293, 1297, 1496, 4463	<code>\l__enumext_write_aux_file_tl</code> . 34, 86, 95, 163 , 2867 , 2873 , 3298 , 3304
<code>\l__enumext_topsep_vii_skip</code> .. 1573, 1582, 1586	<code>enumext*</code> 5, 4899
<code>\l__enumext_topsep_viii_skip</code> . 1595, 1617, 1621	<code>enumXi</code> 577
<code>__enumext_unskip_unkern:</code> .. 36, 234 , 234 , 1346 , 1518 , 3979, 3980, 4020, 4151, 4152, 4169, 5126, 5127, 5401 , 5402	<code>enumXii</code> 577
<code>\l__enumext_vspace_a_star_v_bool</code> 1769	<code>enumXiii</code> 577
<code>\l__enumext_vspace_a_star_vii_bool</code> ... 1791	<code>enumXiv</code> 577
<code>\l__enumext_vspace_a_star_viii_bool</code> ... 1802	<code>enumXv</code> 577
<code>\l__enumext_vspace_a_star_X_bool</code> 91	<code>enumXvi</code> 577
<code>__enumext_vspace_above:</code> 65, 109, 1737 , 1737, 3990	<code>enumXvii</code> 577
<code>__enumext_vspace_above_v:</code> . 66, 1765 , 1765, 4105	<code>enumXviii</code> 577
<code>\l__enumext_vspace_above_v_skip</code> .. 1767, 1771, 1773	Environments provide by enumext :
<code>__enumext_vspace_above_vii:</code> 66, 127, 1787 , 1787, 4956	<code>anskey*</code> 30, 33, 35, 76, 81, 82, 86, 88, 91, 107, 108, 128, 137, 138 , 144 , 146
<code>\l__enumext_vspace_above_vii_skip</code> 1789, 1793, 1795	<code>enumext*</code> 30, 31, 35, 36, 40–44, 47, 49, 50, 52, 53, 55, 62, 63, 66–71, 73, 75–78, 81–84, 86, 87, 89, 93, 94, 100, 101, 104, 106–108, 113, 114, 121, 122, 124, 125, 128–130, 132, 133, 135–139, 141, 145, 148 , 149
<code>__enumext_vspace_above_viii:</code> . 66, 1787 , 1798, 5227	<code>enumext</code> 30, 31, 35, 36, 40–44, 46–58, 61, 63–71, 73, 75–78, 81–84, 86, 87, 89, 93, 94, 98–100, 102, 104, 108, 111, 112, 116, 121, 124, 126, 128, 129, 132, 138, 139, 141, 145 , 146 , 148
<code>\l__enumext_vspace_above_viii_skip</code> 1800, 1804, 1806	<code>keyans*</code> 30, 31, 33–37, 40–43, 47–50, 52, 53, 55, 62, 63, 66, 67, 76, 77, 80, 81, 83, 92, 94, 96, 101, 104, 106, 113, 114, 122, 123, 132, 133, 145, 147 , 149
<code>\l__enumext_vspace_b_star_v_bool</code> 1780	<code>keyanspic</code> 30, 31, 33, 34, 37, 43, 48, 76, 77, 80, 83, 84, 92, 94–96 , 101, 113–118, 120, 147
<code>\l__enumext_vspace_b_star_vii_bool</code> ... 1813	<code>keyans</code> 30, 31, 33, 34, 36, 37, 40, 41, 43, 44, 48–50, 52, 53, 55, 57, 61, 63–66, 76, 77, 80, 81, 83, 84, 92, 94–97 , 101–104 , 111, 113, 115, 117, 120, 124, 133, 145 , 147
<code>\l__enumext_vspace_b_star_viii_bool</code> ... 1824	Environments:
<code>\l__enumext_vspace_b_star_X_bool</code> 91	<code>center</code> 121
<code>__enumext_vspace_below:</code> 66, 110, 1751 , 1751, 4028	<code>description</code> 100, 121
<code>__enumext_vspace_below_v:</code> . 66, 1776 , 1776, 4178	<code>enumerate</code> 121
<code>\l__enumext_vspace_below_v_skip</code> .. 1778, 1782, 1784	<code>flushleft</code> 121
<code>__enumext_vspace_below_vii:</code> 67, 127, 1809 , 1809, 4966	<code>flushright</code> 121
<code>\l__enumext_vspace_below_vii_skip</code> 1811, 1815, 1817	
<code>__enumext_vspace_below_viii:</code> . 67, 1809 , 1820,	

itemize	121	4414, 4423, 4431, 4438, 4443, 4491, 4500, 4590, 4598, 4800, 4911, 4919, 5071, 5180, 5188	
list 35, 39, 50, 89, 100, 104, 109, 111, 113, 115, 117, 121, 124		\IfDocumentMetadataTF .. 496, 512, 526, 539, 3523, 3699, 4864	
lrbox	130	\IfHyperBoolean	387
minipage 35, 39, 41, 42, 55, 58, 59, 115, 116, 118, 120, 121, 124, 131		\IfPackageLoadedT	383
multicols	56–59, 64, 108–110	\IfPackageLoadedTF	7, 395
quotation	121	\ignorespaces .. 1067, 1080, 1092, 1104, 4403, 4916, 4999, 5032, 5055, 5062, 5108, 5128, 5185, 5243, 5277, 5306, 5383, 5403	
quote	121	\inputlineno	281, 295, 308, 316, 324
tabbing	121	int commands:	
trivlist	121	\int_add:Nn	4710, 4759
verbatim	121	\int_case:nn	1241, 1366, 2401, 2427, 2466, 2490
verse	121	\int_case:nnTF	236, 2279
exp commands:		\int_compare:nNnTF .. 560, 776, 790, 807, 814, 1336, 1355, 1509, 1527, 1639, 1658, 1670, 1698, 1899, 1906, 1915, 1926, 1941, 1947, 1958, 2039, 2060, 2081, 2094, 2117, 2133, 2161, 2177, 2208, 2212, 2514, 2520, 3042, 3046, 3050, 3058, 3170, 3174, 3178, 3216, 3236, 3274, 3279, 3284, 3309, 3424, 3874, 3885, 3914, 3927, 3943, 3958, 3973, 4014, 4090, 4094, 4122, 4147, 4163, 4362, 4474, 4478, 4680, 4690, 4706, 4729, 4739, 4755, 4929, 4933, 4977, 4987, 5139, 5151, 5201, 5213, 5418, 5430, 5629, 5761	
\exp_after:wN	5459	\int_compare:nTF	2224, 2233, 2238
\exp_args:Ne 2049, 2112, 2155, 2203, 3208, 3895, 5447		\int_compare_p:nNn	251, 261, 273, 274, 287, 288, 1645, 1646, 2407, 2433, 2815, 2825, 2837, 2838, 2853, 2894, 3924
\exp_args:NV	3003, 3100, 3604, 3622, 3644, 5746	\int_decr:N	4709, 4758
\exp_not:N 43, 617, 743, 786, 800, 848, 1064, 1067, 1078, 1079, 1080, 1091, 1092, 1103, 1104, 2932, 2974, 2975, 3334, 3440, 3441, 5332, 5333, 5456		\int_eval:n .. 366, 902, 2714, 2865, 2975, 3292, 3441, 4053, 4193, 4698, 4747, 4910, 5179, 5333	
\exp_not:n 281, 295, 308, 316, 324, 683, 703, 743, 744, 786, 800, 848, 1065, 1884, 1893, 2542, 2591, 2695, 2708, 2870, 2898, 2908, 2918, 2932, 2933, 3301, 3336, 3338, 4327, 5575, 5585, 5778, 5783		\int_from_alph:n	867, 881
F		\int_from_roman:n	869, 883
\fbox	2598	\int_gadd:Nn	4711, 4760
\fboxrule	2598	\int_gdecr:N	2410, 2415, 2419, 2423, 2436
\fboxsep	2598	\int_gincr:N 2877, 3344, 3460, 3494, 3677, 4003, 4114, 4518, 5005, 5081, 5249, 5315	
file commands:		\int_gset:Nn	442, 472, 2459
\file_if_exist:nTF	3116	\int_gset_eq:NN .. 439, 469, 1962, 1975, 1984, 1992, 2004, 2016, 2025, 2032	
\file_input_stop:	6222	\int_gzero:N .. 337, 338, 339, 1684, 1711, 2252, 2526, 4019, 4168, 5162, 5442	
first	1117	\int_if_exist:NNTF 1921, 1932, 1960, 1990, 2002, 2030, 2153, 2364	
font	636	\int_incr:N 3057, 3426, 3873, 4085, 4361, 4928, 5004, 5200, 5248	
\footnote	40	\int_mod:nn	5153, 5432
\footnote	40, 434, 464	\int_new:N 16, 17, 18, 19, 20, 21, 49, 53, 78, 95, 117, 132, 144, 145, 156, 157, 158, 160, 171, 172, 180, 181, 182, 183, 184, 1923, 1934, 2367	
\footnotemark	444, 474	\int_set:Nn 863, 867, 869, 2112, 2123, 2139, 2145, 2155, 2167, 2183, 2189, 2276, 4584, 4585, 4625, 4656, 4679, 4685, 4701, 4728, 4734, 4750, 5137, 5416, 5625, 5763	
\footnotesize	2975, 3441, 5333	\int_set_eq:NN	3806, 3850, 4708, 4757
\footnotetext	430	\int_sign:n	2461
force-eol	3071	\int_step_function:nnN .. 2127, 2130, 2171, 2174, 2210, 2831, 2844, 2858	
\foreachkeyans	19, 142, 5711	\int_step_function:nnnN	5767
G		\int_step_inline:nn	5677
\getkeyans	19, 137, 5445	\int_step_inline:nnn	4586
group commands:		\int_to_roman:n .. 220, 2112, 2114, 2153, 2155, 2157, 2200, 2205, 2284, 2289, 2294, 2299, 2811, 2848	
\group_begin:	2973, 3018, 3439, 5331, 5514	\int_use:N 359, 364, 365, 1337, 1356, 1671, 2049, 2114, 2125, 2141, 2157, 2169, 2185, 2191, 2204, 3827, 3896,	
\group_end:	2980, 3034, 3447, 5338, 5523		
H			
\hbadness	5137, 5416		
hbox commands:			
\hbox_overlap_left:n	2790, 3506, 5095		
\hbox_set:Nn	606, 4390		
\hbox_set_end:	5136, 5415		
\hbox_set_to_wd:Nnw	5113, 5388		
\hfill 666, 671, 677, 678, 1676, 1703, 2932, 3334, 4802, 4866			
hook commands:			
\hook_gput_code:nnn	5, 208, 212, 216, 379		
\hook_gset_rule:nnnn	380		
\hyperlink	88, 95		
\hyperlink	2932, 3334		
\hypertarget	40		
\hypertarget	405		
I			
\IfDocumentMetadataT 4242, 4250, 4258, 4294, 4302, 4310,			

3944, 3953, 3968, 3974, 4053, 4193, 4683, 4684, 4696, 4732, 4733, 4745, 4910, 5179, 6114, 6118, 6124, 6128, 6205	
\int_zero:N	3429, 5143, 5422
iow commands:	
\iow_char:N	3197, 3198
\iow_close:N	3141
\iow_new:N	112
\iow_now:Nn	3140
\iow_open:Nn	3139
\item	98, 102, 128, 130, 133, 136, 372, 2732, 2738, 2763, 2769, 2891, 3311, 3314, 3512, 3681, 4418, 4419, 4913, 4915, 5182, 5184, 5313
\item*	5, 16, 80, 3679
item-join	2985, 3071
item-pos*	2985, 3071, 3575
item-star	2985, 3071
item-sym*	2985, 3071, 3575
\itemindent	105
\itemindent	104
itemindent	1011
\itemsep	4407
\itemwidth	576, 2598, 4037, 4043, 4182, 4188, 4719, 4723, 4768, 4772
K	
keyans	16, 4195
keyans*	16, 5169
keyanspic	17, 4409
Keys for \anskey provide by enumext:	
break-col	87, 89
force-eol	90
item-join	87, 89
item-pos*	87, 89
item-star	87, 89
item-sym*	87, 89
overwrite	90
write-env	90
Keys for anskey* provide by enumext:	
break-col	87, 89
force-eol	90
item-join	87, 89
item-pos*	87, 89
item-star	87, 89
item-sym*	87, 89
overwrite	90
write-env	90
Keys for environments provide by enumext:	
above*	32, 51, 65, 66, 109, 127
above	32, 51, 65, 66, 109, 127, 133
after	53, 54, 110, 127, 133
align	32, 45, 96, 98, 99, 103, 129, 144
base-fix	51, 68, 83, 107
before*	53, 54, 109, 127, 133
before	53, 54
below*	32, 65-67, 110, 127
below	32, 65-67, 110, 127, 133
check-ans	34, 35, 37, 75-80, 83, 93, 95, 110, 111, 127, 132, 145
columns-sep	55, 108, 131
columns	32, 55, 65, 108
first	53, 54, 131
font	44, 99, 103, 119, 129
item-pos*	98, 100
item-sym*	33, 98, 100

itemindent	32, 52, 98, 102-104, 131
itemsep	50, 106, 131
label-pos	115, 117, 119, 120
label-sep	116
labelsep	44, 105, 129
labelwidth	43, 44, 46-49, 105, 129
label	31, 43, 44, 46, 48, 49, 117, 121
layout-sep	116
layout-sty	116, 120, 121
layout-top	116
lisparindent	106
list-indent	32, 52, 117
list-offset	52, 110, 113
listparindent	52, 131
mark-ans*	80, 83, 97
mark-ans	81, 83, 88
mark-pos*	80, 83, 97
mark-pos	33, 81, 83, 144
mark-ref	81, 83, 85, 88
mark-sep*	80, 83, 96, 97
mark-sep	33, 81, 83, 135
mini-env	32, 40-42, 55, 64, 65, 83, 109, 121, 124, 125, 127, 133
mini-right*	32, 35, 55, 83, 124, 125, 127
mini-right	32, 35, 55, 63, 83, 124, 125, 127
mini-sep	32, 55, 83, 109
mode-box	44, 98-100, 103, 104
no-store	34, 75-77, 83, 89, 92, 98
noitemsep	50
nosep	50
overwrite	33, 91
parindent	106
parsep	50, 106, 117, 131
partopsep	50
ref	31, 46-48, 104, 145
resume*	31, 67, 68, 73, 75, 76, 83, 110, 127, 139
resume	31, 38, 67-73, 75, 76, 83, 110, 127, 139, 140, 150
rightmargin	52, 122
save-ans	33, 38, 68, 69, 72, 73, 75-77, 79, 82-84, 89, 90, 92-95, 102, 111, 118, 129, 132, 133, 135, 137, 138, 140, 145
save-key	33, 68, 82, 83, 107, 127
save-pos	83
save-ref	34, 40, 81, 83, 85, 87, 88, 94, 95, 102, 135
save-sep	80, 83, 94, 134
series	31, 67-69, 71, 73, 83, 107, 110, 127, 140
show-ans	33, 80, 81, 83, 85, 87, 88, 96, 97, 119, 135
show-length	36, 53, 104, 145
show-pos	33, 80, 81, 85, 87, 88, 96, 119, 135
start*	32, 48, 49, 68
start	32, 36, 48, 49, 68
store-key	82
topsep	50, 51, 117
widest	31, 36, 49
wrap-ans*	34, 80, 83, 103, 119
wrap-ans	43, 81, 83, 85, 88
wrap-label*	32, 44, 98, 99, 102, 103, 129, 134
wrap-label	32, 44, 98, 99, 102, 103, 117, 119, 129, 134
wrap-opt	80, 83, 96, 102, 119
wrap-sep	88
write-env	33, 91
keys commands:	
\keys_define:nn	628, 638, 660, 695, 711, 757, 822, 891, 917, 933, 975, 1013, 1036, 1110, 1119, 1198, 1215, 1722, 1833, 1849, 1860, 2312, 2372, 2531, 2566, 2654,

2659, 2985, 3071, 3577, 3593, 3616, 3636, 4316, 5473,
5587, 5703, 5711
\keys_if_exist_p:nn 5699, 5700
\l_keys_key_str 89, 91, 3003, 3100, 3604, 3622, 3644,
5746, 5858
\keys_precompile:nnN .. 138, 203, 203, 5475, 5481,
5487, 5493, 5499, 5505, 5729
\keys_set:nn . 652, 992, 1004, 1221, 1727, 1732, 2049,
2070, 2203, 2217, 2602, 2603, 2607, 2608, 2612, 2613,
2617, 2618, 2622, 2623, 2627, 2628, 3023, 3152, 3887,
3895, 4101, 4334, 4336, 4338, 4340, 4342, 4344, 4346,
4348, 4350, 4352, 4372, 4946, 5222, 5591, 5596, 5597,
5598, 5599, 5602, 5607, 5608, 5609, 5610, 5611, 5612,
5613, 5645, 5755
keyval commands:
\keyval_parse:NNn 1873, 2684, 5563

L

label 709, 755, 822
label-pos 4316
label-sep 4316
Labels provide by enumext:
\Alph* 43, 44
\Roman* 43, 44
\alph* 43, 44
\arabic* 43, 44
\roman* 43, 44
labelsep 636
\labelwidth 44
labelwidth 636
\lastnodetype 236
layout-sep 4316
layout-sty 4316
layout-top 4316
\leftmargin 105
\leftmargin 104, 4402
legacy commands:
\legacy_if:nTF 5066, 5069, 5366, 5369
\legacy_if_gset_false:n 566, 4815
\legacy_if_set_false:n 5068, 5368
\legacy_if_set_true:n 5031, 5054, 5061, 5075, 5276,
5305
\linewidth 109
\linewidth 3998, 4037, 4111, 4182, 4583, 4628, 4659, 4781,
4846
\list 370
list-indent 1011
list-offset 1011
\listparindent 4405
listparindent 1011

M

\makebox 121
\makebox 2792, 3559, 3749, 4508, 4521, 5099, 5378
\makelabel 98, 99, 103, 121
\makelabel 98, 102, 3539, 3555, 3733, 3745
mark-ans 2564, 4316
mark-ans* 2529, 2564
mark-pos 2564, 4316
mark-pos* 2529, 2564
mark-ref 2564
mark-sep 2564, 4316
mark-sep* 2529, 2564
midpenalty 915
mini-env 1196

mini-sep 1196
\minipage 376
\miniright 11, 63, 1637, 1688, 1715, 4017, 4166
mode commands:
\mode_if_math:TF 3066, 3189
\mode_if_vertical:TF 1275, 1303, 1323, 1347, 1498,
1519
\mode_leave_vertical: 990, 1001, 1064, 1078, 2788,
3504, 5093
mode-box 626
msg commands:
\msg_error:nn .. 1690, 1717, 2269, 3027, 3060, 3064,
3156, 3187, 4092, 4096, 4364, 4421, 4476, 4931, 5203,
5215, 5614, 5673
\msg_error:nnn 734, 780, 794, 842, 1641, 1648, 1655,
1686, 1713, 2053, 2074, 2226, 2243, 2303, 2307, 2337,
3009, 3068, 3106, 3168, 3172, 3176, 3180, 3191, 3610,
3628, 3650, 4935, 5208, 5461, 5470, 5556, 5661, 5692,
5701, 5738, 5759
\msg_error:nnnn 3012, 3040, 3044, 3048, 3052, 3109,
3613, 3631, 3653, 4083, 4472, 4480, 5198, 5535, 5741
\msg_error:nnnnn 682, 702, 2541, 2590, 4326
\msg_fatal:nn 3875
\msg_fatal:nnn 580
\msg_info:nnn 9, 12, 385, 397
\msg_line_context: .. 5818, 5823, 5828, 5833, 5862,
5867, 5872, 5887, 5902, 5906, 5910, 5914, 5918, 5922,
5929, 5936, 5942, 5956, 5960, 5965, 5969, 5973, 5977,
5982, 5986, 5990, 5994, 5999, 6046, 6050, 6055, 6060,
6064, 6069, 6145, 6149, 6154, 6159, 6164, 6168, 6172,
6176, 6180, 6184, 6188, 6192, 6196, 6201, 6206
\msg_log:nnn 2356, 2361, 2366
\msg_log:nnnnn 363, 2499, 2504, 2509
\msg_log:nnnnnn 355
\msg_new:nnn 5786, 5790, 5794, 5798, 5803, 5816, 5820,
5825, 5830, 5835, 5844, 5852, 5856, 5860, 5865, 5870,
5885, 5900, 5904, 5908, 5912, 5916, 5920, 5924, 5933,
5939, 5945, 5949, 5953, 5958, 5963, 5967, 5971, 5975,
5980, 5984, 5988, 5992, 5997, 6032, 6036, 6040, 6044,
6048, 6053, 6058, 6062, 6067, 6143, 6147, 6152, 6157,
6162, 6166, 6170, 6174, 6178, 6182, 6186, 6190, 6194,
6198, 6203, 6211
\msg_new:nnnn .. 5807, 6002, 6011, 6020, 6026, 6071,
6081, 6091, 6101, 6111, 6121, 6131, 6137, 6208, 6213,
6216, 6219
\msg_term:nnnn . 2321, 2326, 3816, 3826, 3859, 3864
\msg_term:nnnnn 2480
\msg_warning:nn 4016, 4165
\msg_warning:nnn 3120, 3124, 3129
\msg_warning:nnnn 2517, 2523, 3763, 3768, 4682, 4695,
4731, 4744
\msg_warning:nnnnn 2475, 2485
\multicolsep 108
\multicolsep 1340, 1512, 3964, 4138

N

\NeedsTeXFormat 3
\NewCommandCopy 372
\newcounter 583
\NewDocumentCommand 1637, 2221, 3015, 4468, 5445, 5512,
5621, 5670, 5748
\NewDocumentEnvironment . 3145, 4056, 4195, 4409, 4899,
5169
\newlabel 40
\newlabel 417
no-store 2370

\noindent 4005, 4790, 4855, 5142, 5421

\nointerlineskip 1349, 1352, 1521, 1524, 1678, 1705, 4790, 4855

noitemsep 931

\nopagebreak 1286, 1314, 1349, 1352, 1521, 1524, 1628, 1634

\normalfont 2974, 3440, 5332

nosep 931

O

\obeyedline 3197, 3198

overwrite 3071

P

Packages:

caption 124

enumext . 30, 43, 46, 75, 80, 100, 104, 105, 115, 143, 144

enumitem 43

expl3 121

footnotehyper 39, 41, 42

hyperref 34, 35, 39, 40, 88, 95, 129, 143

latex-lab-block 39

ltxcmd 39, 90

ltsockets 113

lua-visual-debug 58

multicol 30, 143

scontents 90

shortlst 121, 126, 130

tagpdf 113

\par .. 1286, 1314, 1352, 1524, 1628, 1634, 1673, 1678, 1700, 1705, 2940, 3981, 4153, 4171, 4454, 4457, 4603, 4817, 4832, 4878, 4892, 5142, 5421

para commands:

\para_end: 5159, 5439

\parbox 2598

\parindent 5124, 5399

\parsep 56, 117

\parsep 991, 3851, 4386, 4395

parsep 931

\parskip 5125, 5400

\partopsep 3852, 4169, 4406

partopsep 931

peek commands:

\peek_meaning:Ntf 5010, 5024, 5039, 5050, 5254, 5269, 5285

\peek_meaning_remove:Ntf 5017, 5261

\peek_remove_spaces:n 3686

\phantomsection 40

\phantomsection 406

prg commands:

\prg_do_nothing: 410

\prg_new_protected_conditional:Npnn 222, 3112

\prg_replicate:nn 231

\prg_return_false: 226, 3125, 3133

\prg_return_true: 225, 3121, 3130

\printkeyans 19, 138, 5512

prop commands:

\prop_const_from_keyval:Nn 5662

\prop_count:N 357, 2714, 2865, 2977, 3292, 3443, 5335, 5764

\prop_get:NnNtf 5688

\prop_gput_if_not_in:Nnn 2712

\prop_if_exist:Ntf 2354, 5465, 5757

\prop_item:Nn 5467, 5781

\prop_new:N 2357

\ProvidesExplPackage 4

R

\raggedcolumns 3967, 4141

\raisebox 4545

\ref 85, 94

ref 709, 755, 822

\refstepcounter 5078, 5371

regex commands:

\regex_if_match:nnTF 224, 866, 868, 880, 882

\renewcommand 743, 786, 800, 848

\RenewDocumentCommand . 434, 464, 1688, 1715, 3197, 3512, 3539, 3555, 3681, 3733, 3745, 4419

\RequirePackage 13

\resetenumext 74, 2221

resume 1831

resume* 1831

rightmargin 1011

\Roman 44, 48, 49

\Roman 602

\roman 44, 48, 49

\roman 603, 727, 5497

S

save-ans 2310

save-key 2652

save-ref 2564

save-sep 2529, 2564, 4316

scan commands:

\scan_stop: 4418, 4913, 5182, 5456, 5459

seq commands:

\seq_clear:N 5623, 5766

\seq_const_from_clist:Nn 5616

\seq_count:N 358, 4609, 5627

\seq_gclear:N 459, 460, 489, 490

\seq_gput_right:Nn 445, 446, 475, 476, 2721

\seq_if_empty:Ntf 452, 482, 5529, 5641

\seq_if_exist:Ntf 2359, 5527

\seq_if_in:NnNtf 5533

\seq_item:Nn 4596

\seq_map_function:NN 5632

\seq_map_inline:Nn 5542, 5550, 5642, 5643

\seq_map_pairwise_function:NNN 454, 484

\seq_new:N 118, 119, 121, 142, 173, 174, 175, 176, 2362

\seq_pop_left:NN 5631

\seq_put_right:Nn 4482, 5639, 5655, 5776

\seq_set_from_clist:Nn 5624

\seq_set_map_e:NNn 5633

\seq_use:Nn 203, 204, 5772

series 1831

\setcounter .. 877, 881, 883, 4051, 4193, 4451, 4910, 5179

\setenumext 6, 139, 5621

\setenumextmeta 6, 141, 5662

show-ans 2529, 2564, 4316

show-length 1108

show-pos 2529, 2564, 4316

skip commands:

\skip_add:Nn 1246, 1255, 1264, 1277, 1281, 1305, 1309, 1325, 1383, 1385, 1399, 1402, 1423, 1425, 1439, 1442, 1462, 1464, 1478, 1481, 1500, 1549, 1550, 1561, 1563, 4395, 4404

\skip_gset:Nn 1576, 1580, 1584

\skip_gzero_new:N 1571, 1572

\skip_horizontal:N .. 1079, 1091, 1103, 5096, 5108, 5146, 5383, 5425

\skip_horizontal:n .. 1065, 2789, 2797, 3505, 3507, 4526, 4995, 5094, 5128, 5239, 5403

<code>\skip_if_eq:nnTF</code>	1244, 1253, 1262, 1369, 1409, 1449, 1537, 1573, 1595, 1739, 1753, 1767, 1778, 1789, 1800, 1811, 1822
<code>\skip_new:N</code>	72, 73, 74, 79, 80, 81, 82, 83, 84, 195
<code>\skip_set:Nn</code>	1229, 1233, 1291, 1295, 1319, 1372, 1373, 1391, 1412, 1413, 1431, 1451, 1452, 1470, 1494, 1540, 1541, 1555, 1575, 1579, 1597, 1601, 1605, 1611, 1615, 1619, 4379
<code>\skip_set_eq:NN</code>	1330, 1331, 1333, 1340, 1505, 1506, 1507, 1512, 3804, 3848, 3851, 5125, 5400
<code>\skip_sub:Nn</code>	1379, 1381, 1395, 1397, 1419, 1421, 1435, 1437, 1458, 1460, 1474, 1476, 1547, 1548, 1559, 1560
<code>\skip_use:N</code>	1231, 1235, 1279, 1283, 1287, 1307, 1311, 1321, 1327, 1740, 1744, 1747, 1754, 1758, 1761, 3981
<code>\skip_vertical:N</code>	567, 570, 1003, 4816, 4830, 5161, 5441
<code>\skip_vertical:n</code>	1002, 5160, 5440
<code>\skip_zero:N</code>	1339, 1353, 1491, 1492, 1493, 1511, 1525, 3852, 3964, 4138, 4406, 4407
<code>\skip_zero_new:N</code>	1570, 1592, 1593, 1594
<code>\c_zero_skip</code>	567, 570, 1003, 1244, 1253, 1262, 1410, 1449, 1573, 1595, 1740, 1754, 1767, 1778, 1789, 1800, 1811, 1822, 4816, 4830, 5161, 5441
<code>\small</code>	5480, 5486, 5492, 5498, 5504, 5510
<code>\smash</code>	3557, 3747
socket commands:	
<code>\socket_assign_plug:nn</code>	4244, 4252, 4260, 4296, 4304, 4312
<code>\socket_new:nn</code>	4214, 4264
<code>\socket_new_plug:nnn</code>	4215, 4223, 4231, 4265, 4273, 4282
<code>\socket_use:n</code>	4297, 4305, 4313
<code>\socket_use:nn</code>	4245, 4253, 4261
<code>start</code>	889
<code>start*</code>	889
<code>start-list-tags</code>	4214, 4264
<code>\stepcounter</code>	438, 468, 4389, 4538
<code>stop-list-tags</code>	4214, 4264
<code>stop-start-tags</code>	4214, 4264
str commands:	
<code>\c_backslash_str</code>	3068, 5823, 5828, 5833, 5838, 5840, 5842, 5847, 5849, 5947, 5951, 5955, 5965, 5969, 5977, 5978, 5982, 5994, 5995, 5999, 6000, 6021, 6023, 6027, 6029, 6069, 6132, 6134, 6138, 6140, 6149, 6150, 6154, 6159, 6160, 6164, 6168, 6172
<code>\c_circumflex_str</code>	114
<code>\c_colon_str</code>	2864, 3291, 5456
<code>\c_left_brace_str</code>	5928, 5935, 5941
<code>\c_percent_str</code>	114
<code>\c_right_brace_str</code>	5928, 5935, 5941
<code>\str_case:nn</code>	244, 301, 3384
<code>\str_case:nnTF</code>	1880, 1888, 2691, 2699, 5570, 5579
<code>\str_clear:N</code>	3884, 4945
<code>\str_const:Nn</code>	113
<code>\str_count:n</code>	231
<code>\str_if_empty:NTF</code>	1897, 1973, 1980, 2014, 2021
<code>\str_if_eq:nnTF</code>	2264, 2277, 3808, 3855, 5672
<code>\str_if_in:nnTF</code>	5452
<code>\str_new:N</code>	48, 75, 127, 128, 129, 147, 190
<code>\str_set:Nn</code>	667, 673, 679, 698, 699, 700, 2263, 2275, 2537, 2538, 2539, 2586, 2587, 2588, 4321, 4324
<code>\str_set_eq:NN</code>	3407, 5323, 5340
<code>\str_use:N</code>	3561
<code>\strut</code>	3557, 3747
<code>\strutbox</code>	1358, 1361, 1372, 1373, 1384, 1386, 1401, 1404,

	1412, 1413, 1424, 1426, 1441, 1444, 1451, 1452, 1463, 1465, 1480, 1483, 1529, 1532, 1540, 1541, 1549, 1550, 1562, 1564, 1575, 1576, 1579, 1586, 1599, 1607, 1613, 1621, 4398, 4404, 4454, 4462, 4551
--	--

T

tag commands:

<code>\tag_mc_begin:n</code>	4221, 4271, 4280
<code>\tag_mc_begin_pop:n</code>	4237, 4289, 4446, 4448
<code>\tag_mc_end:</code>	4225, 4275, 4284
<code>\tag_mc_end_push:</code>	4218, 4268, 4434
<code>\tag_resume:n</code>	4217, 4267, 4425, 4433, 4502, 4600, 4800, 4864
<code>\tag_struct_begin:n</code>	4219, 4220, 4227, 4228, 4229, 4269, 4270, 4277, 4278, 4279, 4435
<code>\tag_struct_end:n</code>	4226, 4233, 4234, 4235, 4236, 4276, 4285, 4286, 4287, 4288, 4445, 4447, 4919, 5188
<code>\tag_suspend:n</code>	4238, 4290, 4416, 4427, 4440, 4493, 4592, 4911, 5180
<code>\tag_tool:n</code>	4426

TeX and \LaTeX commands:

<code>\auxout</code>	415
<code>\@currentvir</code>	244, 301
<code>\protected@write</code>	415

tex commands:

<code>\tex_scantokens:D</code>	205
--------------------------------	-----

text commands:

<code>\text_expand:n</code>	5448
<code>\textasteriskcentered</code>	2534, 2581
<code>\textborn</code>	3581
<code>\textreferencemark</code>	2569
<code>\thepage</code>	421

tl commands:

<code>\c_space_tl</code>	3363, 3376, 5872, 5887, 5910, 5914, 6113, 6114, 6123, 6124, 6184, 6188, 6206
<code>\tl_clear:N</code>	665, 672, 2084, 2097, 2527, 2638, 2648, 2669, 2677, 2884, 3235, 3308, 5291
<code>\tl_clear_new:N</code>	612
<code>\tl_const:Nn</code>	596
<code>\tl_gclear:N</code>	349, 350, 351, 1943, 1949, 3550, 3570, 4836, 4896, 5097
<code>\tl_gclear_new:N</code>	1917, 1928
<code>\tl_gput_right:Nn</code>	597
<code>\tl_greplace_all:Nnn</code>	618
<code>\tl_gset:Nn</code>	278, 279, 292, 293, 1918, 1929, 1944, 1950, 2353, 3481, 5045
<code>\tl_gset_eq:NN</code>	614, 3477, 5090
<code>\tl_if_blank:nTF</code>	3007, 3025, 3104, 3154, 3608, 3626, 3648, 5088, 5736
<code>\tl_if_empty:NTF</code>	732, 750, 778, 792, 809, 816, 840, 854, 1978, 1982, 2019, 2023, 2086, 2099, 2200, 2214, 2332, 2391, 2728, 2759, 2904, 3218, 3245, 3318, 3356, 3369, 3502, 4607, 5294, 5653
<code>\tl_if_empty:nTF</code>	2041, 2062
<code>\tl_if_exist:NTF</code>	2046, 2067
<code>\tl_if_novalue:nTF</code>	436, 466, 3021, 3150, 3243, 3316, 3349, 3456, 3475, 3483, 3658, 3882, 4370, 4943, 5220, 5292
<code>\tl_map_inline:Nn</code>	615
<code>\tl_new:N</code>	29, 30, 31, 34, 37, 38, 41, 42, 46, 54, 55, 59, 60, 96, 97, 98, 104, 105, 106, 107, 108, 109, 111, 115, 116, 120, 122, 123, 124, 133, 136, 137, 154, 163, 164, 165, 168, 189
<code>\tl_put_left:Nn</code>	2736, 2767, 2889, 4820, 4881, 5310, 5313

<code>\tl_put_right:Nn</code>	. 613, 846, 2740, 2771, 2818, 2828, 2841, 2856, 2862, 2867, 2891, 2896, 2903, 2906, 2916, 2921, 2924, 2930, 3203, 3238, 3241, 3247, 3249, 3276, 3281, 3286, 3289, 3298, 3311, 3314, 3320, 3322, 3332, 5296, 5297
<code>\tl_remove_all:Nn</code> 5652
<code>\tl_remove_once:Nn</code> 2806, 3261
<code>\tl_replace_all:Nnn</code> 617, 3198, 5687
<code>\tl_reverse:N</code> 2805, 2807, 3260, 3262
<code>\tl_set:Nn</code>	. 43, 248, 258, 305, 306, 313, 314, 321, 322, 582, 666, 671, 677, 678, 731, 741, 775, 784, 798, 839, 1062, 1076, 1089, 1101, 2085, 2098, 2331, 2639, 2649, 2670, 2678, 2971, 3090, 3196, 3351, 3437, 3596, 4355, 5299, 5329, 5650, 5686, 5756
<code>\tl_set_eq:NN</code>	. . 623, 737, 783, 797, 845, 2804, 3259, 3272, 3405, 5322
<code>\tl_to_str:n</code> 2046, 2050, 2067, 2071, 5448
<code>\tl_trim_spaces:n</code>	. . . 613, 5639, 5650, 5656, 5672
<code>\tl_use:N</code>	619, 622, 752, 811, 818, 856, 1134, 1138, 1142, 1146, 1150, 1154, 1158, 1162, 1166, 1170, 1174, 1178, 1182, 1186, 1190, 1194, 2794, 2811, 2819, 2830, 2843, 2848, 2859, 3464, 3470, 3498, 3541, 3543, 3549, 3564, 3661, 3665, 3672, 3735, 3738, 3740, 3753, 4063, 4201, 4523, 4531, 4827, 4888, 5101, 5129, 5130, 5380, 5404, 5409, 5516, 5517, 5518, 5519, 5520, 5538, 5635, 5754
token commands:	
<code>\token_to_str:N</code> 417
<code>\topsep</code> 4169, 4404
<code>topsep</code> 931
<code>\topskip</code> 1339, 1511
U	
<code>\unkern</code> 239
<code>unknown</code> 2985, 3071, 3591, 3616, 3634
<code>\unskip</code> 238
use commands:	
<code>\use:N</code> 232, 3546, 3567, 4065
<code>\use:n</code> 1871, 2682, 5454, 5561
<code>\use_none:nn</code> 409, 5693
<code>\usecounter</code> 3807, 3853
V	
<code>\value</code> 1963, 2005, 2017, 2025, 2033
vbox commands:	
<code>\vbox_set:Nn</code> 4495
<code>\vbox_set_top:Nn</code> 4825, 4886
<code>\vspace</code>	991, 1744, 1747, 1758, 1761, 1771, 1773, 1782, 1784, 1793, 1795, 1804, 1806, 1815, 1817, 1826, 1828
W	
<code>widest</code> 889
<code>wrap-ans</code> 2564
<code>wrap-ans*</code> 2529, 2564, 4316
<code>wrap-label</code> 636
<code>wrap-label*</code> 636
<code>wrap-opt</code> 2529, 2564, 4316
<code>write-env</code> 3071