

enumext

ENUMERATE EXERCISE SHEETS

V1.5 2025-06-15^{*}

©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	11
1.2	The concept of left margin	3	6.1.1	Keys for label and ref	12
1.3	User interface	3	6.1.2	Keys for wrap and marks	12
1.3.1	Internal counters	3	6.1.3	Keys for debug and checking	13
1.3.2	Public dimension	3	6.2	The command <code>\anskey</code>	13
1.3.3	Support for 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>	14
1.3.5	The <code>\label</code> and <code>\ref</code> system	4	6.3.1	Keys for <code>anskey*</code>	14
1.3.6	Support for <code>\footnote</code>	4	6.4	The environment <code>keyans</code>	15
2	The environments provided	5	6.4.1	The <code>\item*</code> in <code>keyans</code>	16
2.1	The environment <code>enumext</code>	5	6.5	The environment <code>keyanspic</code>	16
2.2	The environment <code>enumext*</code>	5	6.5.1	Keys for <code>keyanspic</code>	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	18
2.4	The command <code>\item</code> in <code>enumext*</code>	6	6.6.1	The command <code>\getkeyans</code>	18
3	The command <code>\setenumext</code>	6	6.6.2	The command <code>\foreachkeyans</code>	18
4	The command <code>\setenumextmeta</code>	6	6.6.3	The command <code>\printkeyans</code>	19
5	The keyval system	7	7	Full examples	20
5.1	Keys for label and ref	7	8	Tagged PDF examples	23
5.2	Keys for spaces	8	9	The way of non-enumerated lists	23
5.2.1	Vertical spaces	8	10	References	25
5.2.2	Horizontal spaces	9	11	Change history	26
5.3	Keys for add code	9	12	Index of Documentation	27
5.4	Keys for start, series and resume	10	13	Implementation	29
5.5	Keys for multicol	10	14	Index of Implementation	145
5.6	Keys for minipage	11			
5.6.1	The command <code>\miniright</code>	11			
5.6.2	The key <code>mini-right</code>	11			

Motivation and acknowledgments

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

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

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

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

^{*}This file describes a documentation for v1.5, last revised 2025-06-15.

[†]E-mail: pablgonz@educarchile.cl.

License and Requirements

Permission is granted to copy, distribute and/or modify this software under the terms of the LaTeX Project Public License (lpl), version 1.3 or later (<https://www.latex-project.org/lpl.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 / 160

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

1. B) $x = 5$

2. D)

3. C) some note
- ⌘ 4. E) A duck

⌘ 5. D) “other note”

⌘

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

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

1.1 Description and usage

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

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

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

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

The package is loaded in the usual way:

```
\usepackage{enumext}
```

1.2 The concept of left margin

There is a direct relationship between the parameters `\leftmargin`, `\itemindent`, `\labelwidth` and `\labelsep` plus an “extra space” that makes it difficult to obtain the desired horizontal spaces in a `list` environment. Usually we don’t want the `list` to go beyond the left margin of the page, but since these four values are related, that causes a problem.

The `enumitem`[6] package adds the `\labelindent` parameter to solve some of these problems. A simplified representation of this in the figure 1.



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

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



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

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



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

1.3 User interface

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

1.3.1 Internal counters

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

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

1.3.2 Public dimension

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

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

1.3.3 Support for multicol

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

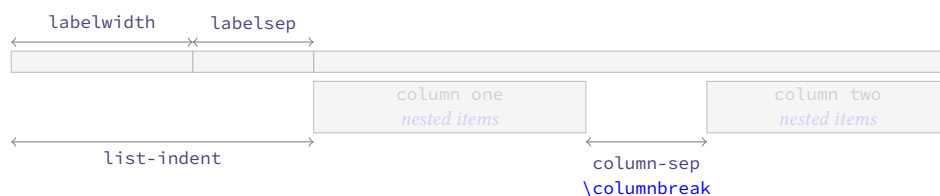


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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

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

1.3.4 Support for minipage

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



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

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

1.3.5 The \label and \ref system

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

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 `(\alph*)`, 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\alph*\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 `\alph*`, ‘VIII’ for `\Roman*` and ‘viii’ for `\roman*`.

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

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

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

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

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

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

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

Wraps the *current* $\langle label \rangle$ defined by `label` key referenced by `\{ \#1 \}` 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 spaces

`show-length = {⟨true⟩|⟨false⟩}` default: *false*

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

5.2.1 Vertical spaces

`topsep = {⟨rubber length⟩|⟨rigid length⟩}` default: *by levels*

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

`parsep = {⟨rubber length⟩|⟨rigid length⟩}` default: *by levels*

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

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

`partopsep = {⟨rubber length⟩|⟨rigid length⟩}` default: *by levels*

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

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

`itemsep = {⟨rubber length⟩|⟨rigid length⟩}` default: *by levels*

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

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

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

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

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

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

`base-fix` *⟨value forbidden⟩* default: *not used*

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

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

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

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

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

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

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

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

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

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

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

5.2.2 Horizontal spaces

`list-offset = { $\langle rigid length \rangle$ }` default: *opt*

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

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

Sets the *indentation* of the whole environment under the box defined by `labelwidth` and `labelsep` keys. Internally sets the value of `\leftmargin` and `\itemindent` for the current level. If `list-indent=opt` is set in the environments `enumext` and `keyans` the $\langle label \rangle$ will be part of the text, separated by the value of the `labelsep` key and the *first word*, in simple terms it will look like a “common paragraph”.

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

`itemindent = { $\langle rigid length \rangle$ }` default: *opt*

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 *opt* and is applied internally using `\hspace` without modifying the value of `\itemindent`.

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

`rightmargin = { $\langle rigid length \rangle$ }` default: *opt*

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 *opt*. Internally sets the value of `\rightmargin` for the current level.

`listparindent = { $\langle rigid length \rangle$ }` default: *opt*

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

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

5.3 Keys for add code

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

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

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

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

`first` = {`<code>`} default: *not used*
 Executes {`<code>`} when “starting” the environment. The {`<code>`} must be passed between braces, is executed right “after” all `list parameters` are done, after the second argument of list, just before the first occurrence of \item: \begin{list}{`<arg one>`}{`<arg two>`}{`<code>`}\item.

- 🔴 Keep in mind that the code set in this key will affect the entire “body” of the environment and therefore the inner levels of the list and the `keyans` environment. It is recommended to set this key per level.
- 🔴 In the `enumext*` and `keyans*` environments this key is executed after the `listparindent`, `parsep` and `itemindent` keys within the `minipage` environment in which the “item content” is placed.

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

5.4 Keys for start, series and resume

`start` = {`<integer | integer expression>`} default: `1`
 Sets the *start value* of the numbering on the current level. The {`<integer expression>`} must be passed between braces, internally is evaluated and pass to the counter defined by `label` key on the current level, i.e. it is equivalent to enter `start={\dimeval{100*\value{chapter}}` or `start={100*\value{chapter}}`.

`start*` = {`<integer | string>`} default: *not used*
 Sets the *start value* of the numbering on the current level. Internally `<string>` is converted and passed as value to the counter defined by `label` key on the current level, i.e. it is equivalent to enter `start=5`, `start=E` or `start=v`.

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

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

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

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

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

5.5 Keys for multicol

`columns` = {`<integer>`} default: `1`
 Set the *number of columns* to be used by the `multicol` environment within the 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` = {`<rigid length>`} default: *by level*
 Set the *space between columns* used by the `multicol` 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`.

5.6 Keys for minipage

`mini-env = {⟨rigid length⟩}`

default: *not used*

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

`mini-sep = {⟨rigid length⟩}`

default: `0.3333em`

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

5.6.1 The command `\miniright`

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

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

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

5.6.2 The key `mini-right`

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

`mini-right = {⟨content⟩}`

default: *not used*

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

`mini-right* = {⟨content⟩}`

default: *not used*

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

6 The storage system

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

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

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

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

6.1 Keys for storage system

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

`save-ans = {⟨store name⟩}`

default: *not set*

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

`save-key = {⟨key list⟩}`

default: *not set*

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

```

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

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

This is a “switch-key” that does not receive an argument and disables the “storing content” in the *sequence* and *prop list* $\{\langle store name \rangle\}$ set by `save-ans` key at the entire level or a nested environment in which it runs. This key is intended for use in internal levels or nested `enumext` or `enumext*` environments in which you want to use `enumext` or `enumext*` but “without” using the `\anskey` command or use `anskey*` environment and “without” interfering with the `check-ans` key.

6.1.1 Keys for label and ref

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

Activates the “internal label and ref” mechanism for referencing “stored content” in *prop list* $\{\langle store name \rangle\}$ set by `save-ans` key. To reference the location of the “stored content” within the environment you must use `\ref{\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 `\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 `\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 = {⟨rigid length⟩}` default: `labelsep`
 Sets the *horizontal space* between the box containing the “symbol” defined by `mark-ans` key and the *mandatory argument* ⟨*content*⟩ passed to the `\anskey` command and the *body* in `anskey*` environment.

`wrap-ans = {⟨code {#1} more code⟩}` default: `\fbox+\parbox{#1}`
 Wraps the *mandatory argument* ⟨*content*⟩ passed to the `\anskey` and the ⟨*body*⟩ in `anskey*` environment referenced by {#1} when using the `show-ans` or `show-pos` keys. The {⟨*code*⟩} must be passed between braces and only affects how the *argument* or *body* is displayed and NOT the “stored content” in the *sequence* and *prop list* {⟨*store name*⟩} set by `save-ans` key. If this key is passed using `\setenumext` it is necessary to use double ‘{#1}’.

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

`mark-ans* = {⟨symbol⟩}` 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* = {⟨left | right | center⟩}` 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* = {⟨rigid length⟩}` default: `labelsep`
 Sets the *horizontal space* between the box containing the “symbol” defined by `mark-ans*` key and the current ⟨*label*⟩ for `\item*` and `\anspic*` in the `keyans`, `keyans*` and `keyanspic` environments.

`wrap-ans* = {⟨code {#1} more code⟩}` default: `not used`
 Wraps the *current* ⟨*label*⟩ 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 {⟨*code*⟩} must be passed between braces and *only* affects how the ⟨*label*⟩ is displayed and NOT the “stored label” in the *sequence* and *prop list* {⟨*store name*⟩} 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 ⟨*label*⟩ to be displayed in red when using `show-ans` you just set `wrap-ans*={\textcolor{red}{#1}}`.

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

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

`break-col` $\langle value forbidden \rangle$ default: *not used*

Stores $\{\langle content \rangle\}$ in the *sequence* $\{\langle store name \rangle\}$ of the form `\columnbreak \item $\langle content \rangle$` .

`item-join` = $\{\langle columns \rangle\}$ default: *not set*

Set the *number of columns* to be used for `\item($\langle columns \rangle$)` and stores $\{\langle content \rangle\}$ in the *sequence* $\{\langle store name \rangle\}$ of the form `\item($\langle columns \rangle$) $\langle content \rangle$` .

`item-star` $\langle value forbidden \rangle$ default: *not used*

Stores $\{\langle content \rangle\}$ in the *sequence* $\{\langle store name \rangle\}$ of the form `\item* $\langle content \rangle$` .

`item-sym*` = $\{\langle symbol \rangle\}$ default: *not set*

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

`item-pos*` = $\{\langle rigid length \rangle\}$ default: *not set*

Sets the *offset* for `\item*` when using the keys `item-star` and `item-sym*` and stores $\{\langle content \rangle\}$ in the *sequence* $\{\langle store name \rangle\}$ of the form `\item*[$\langle symbol \rangle$][$\langle offset \rangle$] $\langle content \rangle$` .

Example

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

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

6.3 The environment anskey*

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

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

By design the environment cannot be nested but full supports “verbatim material” in the $\langle body \rangle$ 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 \LaTeX 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, \LaTeX will return an “error” message if this happens. In a similar way comments “%” or “any character” after `\end{anskey*}` on the same line \LaTeX 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 = { $\langle true \mid false \rangle$ }

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]
    \first answer
  \end{anskey*}

  \item Text containing our instructions or questions.

  \begin{enumext}
    \item Question.
      \begin{anskey*}
        \second answer
      \end{anskey*}
    \end{enumext}

  \item Text containing our instructions or questions.

  \begin{anskey*}
    \third answer
  \end{anskey*}

  \item Text containing our instructions or questions.

  \begin{anskey*}
    \fourth answer
  \end{anskey*}
\end{enumext}
```

★ 5. Text containing our instructions or questions.

[5] First answer with verbatim

6. Text containing our instructions or questions.

(a) Question.

[6] second answer

7. Text containing our instructions or questions.

[7] third answer

8. Text containing our instructions or questions.

[8] fourth answer

6.4 The environments `keyans` and `keyans*`

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

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

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

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

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

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

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

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

6.4.1 The \item* in keyans and keyans*

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

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

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


Example

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

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

  \item Text containing a question and image.

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

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

6.5 The environment keyanspic

`keyanspic` `\begin{keyanspic}[\langle key = val \rangle] \anspic*[\langle content \rangle]{\langle drawing or tabular \rangle} \end{keyanspic}`

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

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

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

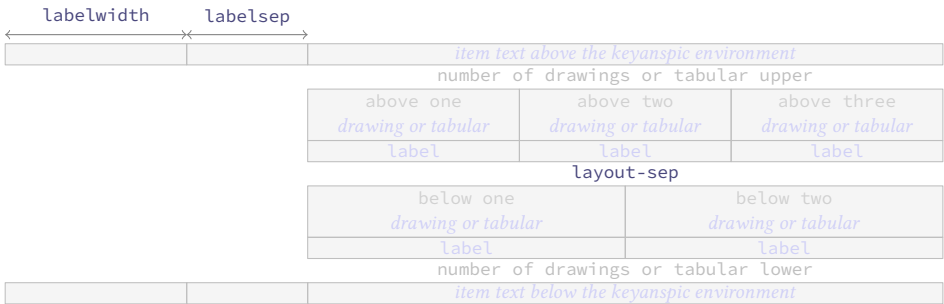


Figure 6: Representation of the `keyanspic` environment with `layout-sty={\langle 3, 2 \rangle}` in `enumext`.

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

6.5.1 Keys for keyanspic

`label-pos = {⟨above | below⟩}`

default: *below*

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

`label-sep = {⟨rubber length | rigid length⟩}`

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 = {⟨n° upper , n° lower⟩}`

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 ⟨*n*° lower⟩ is omitted the *drawings* or *tabular* will be put on a *single line*.

`layout-sep = {⟨rubber length | rigid length⟩}`

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 = {⟨rubber length | rigid length⟩}`

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{⟨drawing or tabular⟩}`
`\anspic*` `\anspic*[⟨content⟩]{⟨drawing or tabular⟩}`

The `\anspic` command take three arguments, the *starred argument* ‘`*`’ store the current ⟨*label*⟩ next to the *optional argument* ⟨*content*⟩ in *sequence* and *prop list* {⟨*store name*⟩} set by `save-ans` key.

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

Example

```
\begin{enumext}[save-ans=test,show-ans=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.



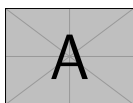
A)



B)



C)

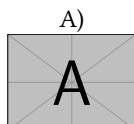


D)

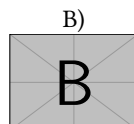


* E) [note]

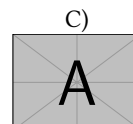
2. Question with images and labels above.



A)



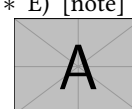
B)



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

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

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

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

6.6.2 The command `\foreachkeyans`

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

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

Options for command

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

Establishes the *separation* between “each” `{<content>}` stored in *prop list* `{<store name>}`. For example, you can use `sep={\[\[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 ⟨keys⟩ 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 ⟨keys⟩ 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 ⟨keys⟩ for the environment `enumext` “saved” in the *sequence* {⟨store name⟩} we will use `\setenumext[⟨print , level⟩]{⟨keys⟩}` and if we need to set the ⟨keys⟩ 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 \LaTeX2e\ 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) ~~TeX~~ze 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 arara¹ tool to compile them.

Example 1

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

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

 36 km/h.
- B

 360 km/h.
- C

 27,8 km/h.
- D

 $3,60 \times 10^8$ km/h.

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

A

 36 km/h.

B

 360 km/h.

C

 27,8 km/h.

D

 $3,60 \times 10^8$ km/h.2. In fisica nucleare si usa l'angstrom (simbolo: $1 \text{ \AA} = 1 \times 10^{-10}$ m) e il fermi o femtometro ($1 \text{ fm} = 1 \times 10^{-15}$ m). Qual è la relazione tra queste due unità di misura?

A

 $1 \text{ \AA} = 1 \times 10^5 \text{ fm}$.

B

 $1 \text{ \AA} = 1 \times 10^{-5} \text{ fm}$.

C

 $1 \text{ \AA} = 1 \times 10^{-15} \text{ fm}$.

D

 $1 \text{ \AA} = 1 \times 10^3 \text{ fm}$.4. In fisica nucleare si usa l'angstrom (simbolo: $1 \text{ \AA} = 1 \times 10^{-10}$ m) e il fermi o femtometro ($1 \text{ fm} = 1 \times 10^{-15}$ m). Qual è la relazione tra queste due unità di misura?

A

 $1 \text{ \AA} = 1 \times 10^5 \text{ fm}$.

B

 $1 \text{ \AA} = 1 \times 10^{-5} \text{ fm}$.

C


 $1 \text{ \AA} = 1 \times 10^{-15} \text{ fm}$.

D

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

1. B
2. A
3. B
4. A

Example 2

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

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

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

A 36 km/h.

✓ B 360 km/h.

C 27,8 km/h.

D $3,60 \times 10^8$ km/h.
2. In fisica nucleare si usa l'angstrom (simbolo: $1 \text{ \AA} = 1 \times 10^{-10}$ m) e il fermi o femtometro ($1 \text{ fm} = 1 \times 10^{-15}$ m). Qual è la relazione tra queste due unità di misura?

✓ A $1 \text{ \AA} = 1 \times 10^5 \text{ fm}$.

B $1 \text{ \AA} = 1 \times 10^{-5} \text{ fm}$.

C $1 \text{ \AA} = 1 \times 10^{-15} \text{ fm}$.

D $1 \text{ \AA} = 1 \times 10^3 \text{ fm}$.
3. La velocità di $1,00 \times 10^2$ m/s espressa in km/h è:

A 36 km/h.

✓ B 360 km/h.

C 27,8 km/h.

D $3,60 \times 10^8$ km/h.
4. In fisica nucleare si usa l'angstrom (simbolo: $1 \text{ \AA} = 1 \times 10^{-10}$ m) e il fermi o femtometro ($1 \text{ fm} = 1 \times 10^{-15}$ m). Qual è la relazione tra queste due unità di misura?

✓ A $1 \text{ \AA} = 1 \times 10^5 \text{ fm}$.

B $1 \text{ \AA} = 1 \times 10^{-5} \text{ fm}$.

C $1 \text{ \AA} = 1 \times 10^{-15} \text{ fm}$.


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

✖ 2. A

✖

3. B

✖ 4. A

✖
- Example 3
- A “simple multiple choice” test .
1. First type of questions

A value

B correct

C value

D value

2. Second type of questions

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

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

Test keys

1. B, $x = 5$

✖ 4. E, A duck

✖

2. D


✖ 5. D, other note


✖


3. C, some note

✖


Example 4

A “simple worksheet” using ducks :) .

 Factor $x^2 - 2x + 1$


 Factor $3x + 3y + 3z$

The following questions need to be cuaqtified :)

 True False

(a) $\alpha > \delta$

(b) \LaTeX is cool?

 Related to Linux

(a) You use linux?

©2024–2025 by Pablo González L


21 / 160

- (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. $(x - 1)^2$ | ⌘ | (b) Yes, dnf | ⌘ |
| 2. $3(x + y + z)$ | ⌘ | (c) i. doesn't exist for now :(| ⌘ |
| 3. (a) False | ⌘ | ii. very good | ⌘ |
| (b) Very True! | ⌘ | iii. obsolete | ⌘ |
| 4. (a) Yes | ⌘ | | |

Example 5

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

<div>1</div> <p>Which choice best describes what happens in the passage?</p> <ul style="list-style-type: none">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.	<div>3</div> <p>Which choice best describes what happens in the passage?</p> <ul style="list-style-type: none">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.
<div>2</div> <p>Which choice best describes what happens in the passage?</p> <ul style="list-style-type: none">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.	<div>4</div> <p>Which choice best describes what happens in the passage?</p> <ul style="list-style-type: none">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
12. sample
- A sample
 - B answer
 - C sample
 - D sample
13. sample
- A sample
 - B sample
 - C sample
 - D answer

10. B (8.5a, KSC)
11. D (9.5a, KSC)

12. B (10.5a, KSC)
13. D (11.5a, KSC)













8 Tagged PDF examples

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

To compile the tests with `lualatex-dev` the packages `multicol`, `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 `multicol` environments using the keys `columns`, `columns-sep`, `mini-env`, `mini-right` and `\miniright` command. Source file  and *tagged* PDF .
- The file `enumext-03.tex` contains the tests for the `enumext` and `keyanspic` environments activated by the `save-ans` key together with the `save-sep` and `save-ref` keys and the `\printkeyans` command. Source file  and *tagged* PDF .
- The file `enumext-04.tex` contains the tests for the `\anskey` command and the `anskey*` environment activated by the `save-ans` key along with the `\getkeyans` and `\printkeyans` commands. Source file  and *tagged* PDF .
- The file `enumext-05.tex` contains the tests for the environments `keyans`, `keyans*` and `keyanspic` activated by the key `save-ans` together with the keys `no-store` and `show-ans` and the commands `\setenumext`, `\setenumextmeta`, `\printkeyans` and `\foreachkeyans`. Source file  and *tagged* PDF .
- The file `enumext-06.tex` contains the tests for the environments `enumext` and `enumext*` for *fake itemize* and *description*. Source file  and *tagged* PDF .

9 The way of non-enumerated lists

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

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

Fake itemize environment

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

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

Fake description environment

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

SomeThing A short one-line description.

This is an entry *without* a label.

Something A short *one-line* description text.

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

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

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

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

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

Description indented by label

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

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

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

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

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

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

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

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

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

Description with multi-line labels

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

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

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

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

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

Final notes

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

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

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

Why list environments?

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

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

Why not random questions and other utilities

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

Why has it taken so long?

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

10 References

- [1] HIRSCHHORN, PHILIP. “Using the exam document class”. Available from CTAN, <https://www.ctan.org/pkg/exam>, 2023.
- [2] NIEDERBERGER, CLEMENS. “xsim – eXercise Sheets IMproved”. Available from CTAN, <https://www.ctan.org/pkg/xsim>, 2023.
- [3] MITTELBACH, FRANK. “An environment for multicolumn output”. Available from CTAN, <https://www.ctan.org/pkg/multicol>, 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 $\text{\LaTeX}3$ 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 \LaTeX Project. “`latex-lab` – \LaTeX laboratory”. Available from CTAN, <https://www.ctan.org/pkg/latex-lab>, 2025.
- [20] MITTELBACH, FRANK. “ \LaTeX ’s socket management”. Available from CTAN, <https://ctan.org/tex-archive/macros/latex/base>, 2025.

11 Change history

- | | |
|--------------------------------|--|
| v1.5 (ctan), 2025-06-15 | – Replacing <code>\regex_match:</code> (deprecated) with <code>\regex_if_match:</code> . |
| v1.4 (ctan), 2025-06-09 | – Improved implementation of the <code>start</code> key for <i>tagged</i> PDF. |
| | – Improved implementation of the <code>ref</code> key. |
| | – Fixed the behavior of the <code>save-sep</code> key. |
| | – Fixed the behavior of the <code>resume*</code> key. |
| v1.3 (ctan), 2025-06-01 | – Removed dependency on the <code>scontents</code> package. |
| | – The <code>anskey*</code> environment has been rewritten using the new <code>c</code> -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 <code>font</code> and <code>base-fix</code> 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	10	item-pos*	14
Commands provide by enumext:		item-star	14
\anskey	11-14	item-sym*	14
\anspic	11-13, 16, 17	Keys for \foreachkeyans provide by enumext:	
\foreachkeyans	18	after	19
\getkeyans	13, 18	before	19
\item*	5-7, 11-13, 15, 16	sep	18
\item	5-7, 10, 11, 13, 15, 16	start	18
\miniright	11	step	18
\printkeyans	6, 12, 19	stop	18
\setenumextmeta	6	wrapper	19
\setenumext	5-7, 11, 13, 15, 19	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	14
enumXvi	4	write-env	14
enumXv	4	Keys for environments provide by enumext:	
E		above*	9
Environments provide by enumext:		above	8, 9
anskey*	11-14, 23	after	10
enumext*	4-16, 19, 23	align	7, 13, 23, 24
enumext	4-16, 19, 23	base-fix	8
keyans*	4-15, 23	before*	9, 10
keyanspic	4, 7, 8, 11-14, 16, 23	before	9
keyans	4-17, 23	below*	9
Environments:		below	9
Verbatim	15	check-ans	12, 13
center	5	columns-sep	4, 10, 23
description	5, 23, 24	columns	4, 9, 10, 23
enumerate	1, 3, 5, 25	first	10
figure	5	font	7, 12, 13
flushleft	5	item-pos*	5, 6
flushright	5	item-sym*	5, 6
itemize	5, 23	itemindent	8-10
list	3, 5, 9, 25	itemsep	8
minipage	3-5, 8-11, 23, 25	label-pos	17
multicols	3, 4, 10, 23	label-sep	17
quotation	5	labelsep	3-7, 9, 10, 23, 24
quote	5	labelwidth	3, 4, 6, 7, 9, 10, 12, 13, 23, 24
shortenurerate	5	labelwith	5
tabbing	5	label	7, 8, 10, 15, 16, 23, 24
table	5	labewdith	9
tasks	5	layout-sep	17
trivlist	5	layout-sty	16, 17
verbatim	5	layout-top	17
verse	5	list-indent	3, 9
		list-offset	3, 8, 9, 24

listparindent 9, 10

mark-ans* 12, 13, 15, 17

mark-ans 12, 13

mark-pos* 13, 15, 17

mark-pos 12

mark-ref 12

mark-sep* 13, 15, 17

mark-sep 12, 13

mini-env 4, 9, 11, 23

mini-right* 7, 11

mini-right 7, 11, 23

mini-sep 4, 11

mode-box 7

no-store 11–14, 23

noitemsep 8

nosep 8, 23

overwrite 14

parsep 8, 10, 17

partopsep 8

ref 4, 8, 23

resume* 7, 10, 11

resume 7, 10, 11

rightmargin 9

save-ans 4, 6, 10–19, 23

save-key 10–12, 19

save-ref 4, 7, 12–14, 18, 23

save-sep 12, 15, 17, 23

series 7, 10, 11

show-ans 12, 13, 15, 17, 23

show-length 8

show-pos 12, 13, 15, 17, 18

start* 10

start 10

topsep 8, 9, 17

widest 7

wrap-ans* 13, 15, 17

wrap-ans 12, 13

wrap-label* 7, 24

wrap-label 7, 12, 13, 23, 24

wrap-opt 12, 13, 15, 17

write-env 14

L

\label 4

Labels provide by enumext:

 \Alph* 7, 8, 15

 \Roman* 7, 8

 \alph* 7, 8

 \arabic* 7, 8

 \roman* 7, 8

\labelsep 3, 7

\labelwidth 3, 7

\linewidth 11

\listparindent 9

P

Packages:

 enumerate 25

 enumext 1–5, 7, 12, 16, 23, 25

 enumitem 3, 4, 24, 25

 fancyvrb 15

 footnotehyper 5

 geometry 23

 graphicx 23

 hyperref 4, 5, 12–14, 23, 25

 l3keys 7

 l3prop 25

 l3seq 25

 luamml 23

 multicol 1, 2, 4, 23, 25

 scontents 26

 shortlst 5

 tasks 5

 task 6

 unicode-math 23

 xsim 2

\parsep 8

\partopsep 8

R

\raggedcolumns 4

\ref 4

\rightmargin 9

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} {2025-06-15} {1.5} {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}[2024-09-14]
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.)

```

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

```

Internal variables used by `resume`, `resume*` and `series` keys (§13.25).

```

46 \int_new:N \g__enumext_resume_int
47 \int_new:N \g__enumext_resume_vii_int
48 \tl_new:N \l__enumext_resume_name_tl
49 \bool_new:N \l__enumext_resume_active_bool
50 \tl_new:N \g__enumext_standar_series_tl
51 \tl_new:N \g__enumext_starred_series_tl

```

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

```

\l__enumext_current_widest_dim
\g__enumext_counter_styles_tl
\g__enumext_widest_label_tl
\l__enumext_label_width_by_box

```

The variable `\l__enumext_current_widest_dim` stores the current label width, the variable `\g__enumext_counter_styles_tl` stores the default *⟨label style⟩* and the variable `\g__enumext_widest_label_tl` the label width. These variables are used by `widest` (§13.15) and `label` (§13.13) keys.

```

52 \dim_new:N \l__enumext_current_widest_dim
53 \tl_new:N \g__enumext_counter_styles_tl
54 \tl_new:N \g__enumext_widest_label_tl
55 \box_new:N \l__enumext_label_width_by_box

```

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

```

\l__enumext_leftmargin_tmp_X_bool
\l__enumext_leftmargin_tmp_X_dim
\l__enumext_leftmargin_X_dim
\l__enumext_itemindent_X_dim

```

The boolean variable `\l__enumext_leftmargin_tmp_X_bool` and the dimensional variable `\l__enumext_leftmargin_tmp_X_dim` are used by the `list-indent` key (§13.18). The variables `\l__enumext_leftmargin_X_dim` and `\l__enumext_itemindent_X_dim` are used and set by the function `__enumext_calc_hspace:NNNNNNNNNN` (§13.38.1).

```

56 \cs_set_protected:Npn \__enumext_tmp:n #1
57 {
58   \bool_new:c { \l__enumext_leftmargin_tmp_#1_bool }
59   \dim_new:c { \l__enumext_leftmargin_tmp_#1_dim }
60   \dim_new:c { \l__enumext_leftmargin_#1_dim }

```

```

61     \dim_new:c { \__enumext_itemindent_#1_dim      }
62   }
63   \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }

```

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

```

\l__enumext_multicols_above_X_skip
\l__enumext_multicols_below_X_skip
\g__enumext_multicols_right_X_skip
\l__enumext_align_label_pos_X_str

```

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

```

64   \cs_set_protected:Npn \__enumext_tmp:n #1
65   {
66     \skip_new:c { \__enumext_multicols_above_#1_skip }
67     \skip_new:c { \__enumext_multicols_below_#1_skip }
68     \skip_new:c { \g__enumext_multicols_right_#1_skip }
69     \str_new:c { \__enumext_align_label_pos_#1_str }
70   }
71   \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

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

```

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

```

Internal variables used by `\miniright` command (§13.23.4) and the keys `mini-right`, `mini-right*`, `mini-env` and `mini-sep` (§13.21, §13.23).

```

72   \int_new:N \g__enumext_minipage_stat_int
73   \skip_new:N \l__enumext_minipage_temp_skip
74   \skip_new:N \l__enumext_minipage_left_skip
75   \skip_new:N \l__enumext_minipage_right_skip
76   \skip_new:N \l__enumext_minipage_after_skip
77   \skip_new:N \g__enumext_minipage_right_skip
78   \skip_new:N \g__enumext_minipage_after_skip
79   \cs_set_protected:Npn \__enumext_tmp:n #1
80   {
81     \dim_new:c { \__enumext_minipage_left_#1_dim      }
82     \bool_new:c { \__enumext_minipage_active_#1_bool }
83   }
84   \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }

```

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

```

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

```

The bool vars `\l__enumext_wrap_label_X_bool` and `\l__enumext_wrap_label_opt_X_bool` are used by `wrap-label` and `wrap-label*` keys (§13.13), the integer `\l__enumext_start_X_int` are used by the `start` and `start*` keys (§13.15), the token list `\l__enumext_fake_item_indent_X_tl` is used by `itemindent` key (§13.18.1), the variables `\l__enumext_label_fill_left_X_tl` and `\l__enumext_label_fill_right_X_tl` are used by the `align` key (§13.13). The boolean vars `\l__enumext_vspace_a_star_X_bool`, `\l__enumext_vspace_b_star_X_bool` are used by `above`, `above*`, `below` and `below*` keys (§13.20).

```

85   \cs_set_protected:Npn \__enumext_tmp:n #1
86   {
87     \bool_new:c { \__enumext_wrap_label_#1_bool      }
88     \bool_new:c { \__enumext_wrap_label_opt_#1_bool }
89     \int_new:c { \__enumext_start_#1_int              }
90     \tl_new:c { \__enumext_fake_item_indent_#1_tl    }
91     \tl_new:c { \__enumext_label_fill_left_#1_tl     }
92     \tl_new:c { \__enumext_label_fill_right_#1_tl    }
93     \bool_new:c { \__enumext_vspace_a_star_#1_bool   }
94     \bool_new:c { \__enumext_vspace_b_star_#1_bool   }
95   }
96   \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 (§13.26.1) activates all the mechanism related to `\anskey`, `anskey*`, `keyans`, `keyans*` and `keyanspic` environments.

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

The variables `\l__enumext_store_current_label_tl` and `\l__enumext_store_current_opt_arg_tl` save the *current label* and *optional argument* of `\item*` (§13.37) and `\anspic*` (§13.42.2) for the `keyans`, `keyans*` and `keyanspic` environments.

```

97   \bool_new:N \l__enumext_store_active_bool
98   \tl_new:N \l__enumext_store_name_tl
99   \tl_new:N \g__enumext_store_name_tl
100  \tl_new:N \l__enumext_store_current_label_tl
101  \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` (§13.30) and the variables `\l__enumext_store_anskey_env_tl` save the (*body*) of the environment `anskey*` (§13.31).

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.

```
102 \tl_new:N \l__enumext_store_anskey_arg_tl
103 \tl_new:N \l__enumext_store_anskey_env_tl
104 \bool_new:N \l__enumext_write_anskey_env_bool
105 \tl_new:N \l__enumext_write_anskey_env_file_name_tl
106 \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 (*forced space*) added by \TeX when recording content in a macro. This *string* contains the *reserved phrase* “`%^^Aenumextheol%`” which is added to the end of the argument stored in *sequence* and *prop list* when the key `force-eol` is false.

```
107 \str_const:Nc \c__enumext_anskey_env_hidden_space_str
108 { \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` (§13.48).

```
109 \tl_new:N \l__enumext_setkey_tmpa_tl
110 \tl_new:N \l__enumext_setkey_tmpb_tl
111 \int_new:N \l__enumext_setkey_tmpa_int
112 \seq_new:N \l__enumext_setkey_tmpa_seq
113 \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 (§13.47) and `\foreachkeyans` command (§13.50).

```
114 \tl_new:N \l__enumext_meta_path_tl
115 \seq_new:N \l__enumext_foreach_print_seq
116 \tl_new:N \l__enumext_foreach_name_prop_tl
117 \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_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
\l__enumext_item_symbol_aux_tl
\l__enumext_print_keyans_X_tl
\l__enumext_store_save_key_X_tl
\l__enumext_store_save_key_X_bool
\l__enumext_store_upper_level_X_bool
```

Internal variables used by command `\printkeyans` (§13.47), `show-pos`, `show-ans`, `mark-pos`, `mark-sep` keys (§13.27), `item-sym*` key (§13.35), `save-key` key (§13.27.3) and “*storing structure*”.

```
118 \tl_new:N \l__enumext_print_keyans_starred_tl
119 \bool_new:N \l__enumext_print_keyans_star_bool
120 \str_new:N \l__enumext_mark_position_str
121 \str_new:N \l__enumext_mark_position_v_str
122 \str_new:N \l__enumext_mark_position_viii_str
123 \dim_new:N \l__enumext_mark_sep_tmpa_dim
124 \dim_new:N \l__enumext_mark_sep_tmpb_dim
125 \int_new:N \l__enumext_show_pos_tmp_int
126 \tl_new:N \l__enumext_item_symbol_aux_tl
127 \cs_set_protected:Npn \l__enumext_tmp:n #1
128 {
129   \tl_new:c { \l__enumext_print_keyans_#1_tl }
130   \tl_new:c { \l__enumext_store_save_key_#1_tl }
131   \bool_new:c { \l__enumext_store_save_key_#1_bool }
132   \bool_new:c { \l__enumext_store_upper_level_#1_bool }
133 }
134 \clist_map_inline:nn { i, ii, iii, iv, vii } { \l__enumext_tmp:n {#1} }
```

(End of definition for \l__enumext_print_keyans_starred_tl and others.)

```
\l__enumext_anspic_args_seq
\l__enumext_anspic_mini_width_dim
\l__enumext_anspic_above_int
\l__enumext_anspic_below_int
\l__enumext_anspic_label_above_bool
\l__enumext_anspic_mini_pos_str
\l__enumext_anspic_label_box
\l__enumext_anspic_body_box
\l__enumext_anspic_label_htdp_dim
\l__enumext_anspic_body_htdp_dim
```

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

```
135 \seq_new:N \l__enumext_anspic_args_seq
136 \dim_new:N \l__enumext_anspic_mini_width_dim
137 \int_new:N \l__enumext_anspic_above_int
138 \int_new:N \l__enumext_anspic_below_int
139 \bool_new:N \l__enumext_anspic_label_above_bool
140 \str_new:N \l__enumext_anspic_mini_pos_str
141 \box_new:N \l__enumext_anspic_label_box
142 \box_new:N \l__enumext_anspic_body_box
143 \dim_new:N \l__enumext_anspic_label_htdp_dim
144 \dim_new:N \l__enumext_anspic_body_htdp_dim
```

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

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

```

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

```

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

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

```

154 \bool_new:N \l__enumext_hyperref_bool
155 \bool_new:N \l__enumext_footnotes_key_bool

```

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

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

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

```

156 \tl_new:N \l__enumext_newlabel_arg_one_tl
157 \tl_new:N \l__enumext_newlabel_arg_two_tl
158 \tl_new:N \l__enumext_write_aux_file_tl
159 \cs_set_protected:Npn \__enumext_tmp:n #1
160 {
161   \tl_new:c { l__enumext_label_copy_#1_tl }
162 }
163 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }

```

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

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

```

164 \int_new:N \g__enumext_footnote_standar_int
165 \int_new:N \g__enumext_footnote_starred_int
166 \seq_new:N \g__enumext_footnote_standar_arg_seq
167 \seq_new:N \g__enumext_footnote_starred_arg_seq
168 \seq_new:N \g__enumext_footnote_standar_int_seq
169 \seq_new:N \g__enumext_footnote_starred_int_seq

```

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

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

```

170 \cs_set_protected:Npn \__enumext_tmp:n #1
171 {
172   \bool_new:c { l__enumext_item_starred_#1_bool }
173   \int_new:c { l__enumext_item_column_pos_#1_int }
174   \int_new:c { g__enumext_item_count_all_#1_int }
175   \int_new:c { l__enumext_joined_item_#1_int }
176   \int_new:c { l__enumext_joined_item_aux_#1_int }
177   \int_new:c { l__enumext_tmpa_#1_int }
178   \dim_new:c { l__enumext_tmpa_#1_dim }
179   \box_new:c { l__enumext_item_text_#1_box }
180   \dim_new:c { l__enumext_joined_width_#1_dim }
181   \dim_new:c { l__enumext_item_width_#1_dim }
182   \tl_new:c { g__enumext_item_symbol_aux_#1_tl }
183   \str_new:c { l__enumext_align_label_#1_str }
184   \bool_new:c { g__enumext_minipage_active_#1_bool }
185   \box_new:c { l__enumext_miniright_code_#1_box }
186   \bool_new:c { g__enumext_minipage_center_#1_bool }
187   \dim_new:c { g__enumext_minipage_right_#1_dim }

```

```

188     \skip_new:c { g__enumext_minipage_right_#1_skip }
189   }
190   \clist_map_inline:nn { vii, viii } { \__enumext_tmp:n {#1} }

```

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

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

```

191   \clist_const:Nn \c__enumext_all_envs_clist
192   {
193     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
194     {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
195   }

```

(End of definition for `\c__enumext_all_envs_clist`.)

13.5 Some utility functions

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

```

196   \cs_generate_variant:Nn \keys_precompile:nnN { neN }
197   \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:nn` 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?](#).

```

198   \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.

```

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

```

(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.

```

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

```

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

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

```

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

```

(End of definition for `__enumext_level:`.)

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

```

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

```


(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.

```
221 \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
222 {
223   *~#2
224   \prg_replicate:nn { 14 - \str_count:n {#2} } {~}
225   =~\use:c { #1_use:c } { \__enumext_#2_#3_#1 } \\
226 }
```

(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.

```
227 \cs_new_protected:Nn \__enumext_unskip_unkern:
228 {
229   \int_case:nnT { \lastnodetype }
230   {
231     { 11 } { \unskip }
232     { 12 } { \unkern }
233   }
234 }
```

(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:`

```
235 \cs_new_protected:Nn \__enumext_is_not_nested:
236 {
237   \str_case:en { \@currentenvir }
238   {
239     {enumext}
240     {
241       \tl_set:Nn \l__enumext_envir_name_tl { enumext }
242       \bool_lazy_and:nnT
243       { \bool_not_p:n { \g__enumext_standar_bool } }
244       { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
245       {
246         \bool_gset_true:N \g__enumext_standar_bool
247       }
248     }
249     {enumext*}
250     {
251       \tl_set:Nn \l__enumext_envir_name_tl { enumext* }
252       \bool_lazy_and:nnT
253       { \bool_not_p:n { \g__enumext_starred_bool } }
254       { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
255       {
256         \bool_gset_true:N \g__enumext_starred_bool
257       }
258     }
259   }
260 }
```

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

```
261 \cs_new_protected:Nn \__enumext_is_on_first_level:
262 {
263   \bool_lazy_all:nT
264   {
265     { \bool_if_p:N \g__enumext_standar_bool }
266     { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
267     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
268   }
269   {
```

```

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

```

(End of definition for \l__enumext_is_not_nested: and \l__enumext_is_on_first_level:.)

\l__enumext_keyans_name_and_start:

The function \l__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 `\l__enumext_check_starred_cmd:n` function.

```

292 \cs_new_protected:Nn \l__enumext_keyans_name_and_start:
293 {
294     \str_case:en { \@currenvir }
295     {
296         {keyans}
297         {
298             \tl_set:Nn \l__enumext_envir_name_tl { keyans }
299             \tl_set:Ne \l__enumext_check_start_line_env_tl
300             {
301                 in~'keyans'~start~on~line~\exp_not:V \inputlineno
302             }
303         }
304         {keyans*}
305         {
306             \tl_set:Nn \l__enumext_envir_name_tl { keyans* }
307             \tl_set:Ne \l__enumext_check_start_line_env_tl
308             {
309                 in~'keyans*'~start~on~line~\exp_not:V \inputlineno
310             }
311         }
312         {keyanspic}
313         {
314             \tl_set:Nn \l__enumext_envir_name_tl { keyanspic }
315             \tl_set:Ne \l__enumext_check_start_line_env_tl
316             {
317                 in~'keyanspic'~start~on~line~\exp_not:V \inputlineno
318             }
319         }
320     }
321 }

```

(End of definition for \l__enumext_keyans_name_and_start:.)

13.5.2 Utilities for log and terminal

\l__enumext_reset_global_vars:

The function \l__enumext_reset_global_vars: will be passed to the function \l__enumext_execute_after_env: and will return the global variables to their default values after being used.

\l__enumext_reset_global_int:

\l__enumext_reset_global_bool:

\l__enumext_reset_global_tl:

```

322 \cs_new_protected:Nn \l__enumext_reset_global_vars:
323 {
324     \l__enumext_reset_global_int:
325     \l__enumext_reset_global_bool:
326     \l__enumext_reset_global_tl:
327 }
328 \cs_new_protected:Nn \l__enumext_reset_global_int:

```

```

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

```

(End of definition for __enumext_reset_global_vars: and others.)

__enumext_log_global_vars: 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.

```

346 \cs_new_protected:Nn \__enumext_log_global_vars:
347 {
348   \msg_log:nneeee { enumext } { prop-seq-int-hook }
349   { \g__enumext_store_name_tl }
350   { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
351   { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
352   { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
353 }

```

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.

```

354 \cs_new_protected:Nn \__enumext_log_answer_vars:
355 {
356   \msg_log:nneeee { enumext } { item-answer-hook }
357   { \int_use:N \g__enumext_item_number_int }
358   { \int_use:N \g__enumext_item_anskey_int }
359   { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int } }
360 }

```

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

13.6 Copying list and minipage environments

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

```

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

```

And `minipage` environment provided by \TeX has the following (simplified) plain form:

```

\minipage[⟨pos⟩][⟨height⟩][⟨inner-pos⟩]{⟨width⟩}
  ⟨internal implement⟩
\endminipage

```

As a precaution we copy them using __enumext_at_begin_document:n in case any package redefines the `list` environment or a related command.

🔍 For compatibility with *tagged* PDF we should use \NewCommandCopy and not \cs_new_eq:NN for \item. When *tagged* PDF is active \item is redefined using `ltxcmd` (see `latex-lab-block`[19]).

```

\__enumext_start_list:nn
  \__enumext_stop_list:
  \__enumext_item_std:w
  \__enumext_minipage:w
  \__enumext_endminipage:
361 \__enumext_at_begin_document:n
362 {
363   \cs_new_eq:NN \__enumext_start_list:nn \list
364   \cs_new_eq:NN \__enumext_stop_list: \endlist
365   \NewCommandCopy \__enumext_item_std:w \item
366 }

```

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.

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

```

367 \__enumext_at_begin_document:n
368 {
369     \cs_new_eq:NN \__enumext_minipage:w \minipage
370     \cs_new_eq:NN \__enumext_endminipage: \endminipage
371 }

```

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

13.7 Compatibility with hyperref and footnotehyper

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

```

\__enumext_after_hyperref:
\__enumext_hypertarget:nn
\__enumext_phantomsection:

```

```

372 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
373 \hook_gset_rule:nnnn { begindocument } { enumext } { after } { hyperref }

```

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

```

374 \cs_new_protected:Nn \__enumext_after_hyperref:
375 {
376     \IfPackageLoadedT { hyperref }
377     {
378         \msg_info:nnn { enumext } { package-load } { hyperref }
379         \bool_set_true:N \l__enumext_hyperref_bool
380         \IfHyperBoolean{hyperfootnotes}
381         {
382             \bool_set_true:N \l__enumext_footnotes_key_bool
383         }
384     }
385 }

```

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`.

```

386 \bool_if:NT \l__enumext_footnotes_key_bool
387 {
388     \IfPackageLoadedTF { footnotehyper }
389     {
390         \msg_info:nnn { enumext } { package-load } { footnotehyper }
391     }
392     {
393         \bool_set_false:N \l__enumext_footnotes_key_bool
394     }
395 }

```

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.

```

396 \bool_if:NTF \l__enumext_hyperref_bool
397 {
398     \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
399     \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
400 }
401 {
402     \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
403     \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
404 }
405 }

```

(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.

```

406 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
407 {

```

```

408 \protected@write \auxout { }
409 {
410   \token_to_str:N \newlabel {#1}
411   {
412     {#2}
413     \bool_if:NT \l__enumext_hyperref_bool
414     { { \thepage } {#2} {#1} }
415     { }
416   }
417 }
418 \__enumext_hypertarget:nn {#1} { }
419 \__enumext_phantomsection:
420 }

```

(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.

```

421 \cs_new_protected:Nn \__enumext_footnotetext:nn
422 {
423   \footnotetext[#1]{#2}
424 }
425 \cs_new_protected:Nn \__enumext_renew_footnote:
426 {
427   \RenewDocumentCommand \footnote { o +m }
428   {
429     \tl_if_novalue:nTF {##1}
430     {
431       \stepcounter{footnote}
432       \int_gset_eq:Nc \g__enumext_footnote_standar_int { c@footnote }
433     }
434     {
435       \int_gset:Nn \g__enumext_footnote_standar_int { ##1 }
436     }
437     \footnotemark [ \g__enumext_footnote_standar_int ]
438     \seq_gput_right:Nn \g__enumext_footnote_standar_arg_seq { ##2 }
439     \seq_gput_right:NV
440     \g__enumext_footnote_standar_int_seq \g__enumext_footnote_standar_int
441   }
442 }
443 \cs_new_protected:Nn \__enumext_print_footnote:
444 {
445   \seq_if_empty:NF \g__enumext_footnote_standar_int_seq
446   {
447     \seq_map_pairwise_function:NNN
448     \g__enumext_footnote_standar_int_seq
449     \g__enumext_footnote_standar_arg_seq
450     \__enumext_footnotetext:nn
451   }
452   \seq_gclear:N \g__enumext_footnote_standar_arg_seq
453   \seq_gclear:N \g__enumext_footnote_standar_int_seq
454 }

```

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.

```

455 \cs_new_protected:Nn \__enumext_renew_footnote_mini:
456 {
457   \RenewDocumentCommand \footnote { o +m }
458   {
459     \tl_if_novalue:nTF {##1}
460     {
461       \stepcounter{footnote}
462       \int_gset_eq:Nc \g__enumext_footnote_starred_int { c@footnote }
463     }
464     {
465       \int_gset:Nn \g__enumext_footnote_starred_int { ##1 }

```

```

466     }
467     \footnotemark [ \g__enumext_footnote_starred_int ]
468     \seq_gput_right:Nn \g__enumext_footnote_starred_arg_seq { ##2 }
469     \seq_gput_right:NV
470     \g__enumext_footnote_starred_int_seq \g__enumext_footnote_starred_int
471   }
472 }
473 \cs_new_protected:Nn \__enumext_print_footnote_mini:
474 {
475   \seq_if_empty:NF \g__enumext_footnote_starred_int_seq
476   {
477     \seq_map_pairwise_function:NNN
478     \g__enumext_footnote_starred_int_seq
479     \g__enumext_footnote_starred_arg_seq
480     \__enumext_footnotetext:nn
481   }
482   \seq_gclear:N \g__enumext_footnote_starred_arg_seq
483   \seq_gclear:N \g__enumext_footnote_starred_int_seq
484 }

```

(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.

```

485 \cs_new_protected:Nn \__enumext_renew_footnote_standar:
486 {
487   \bool_if:NT \g__enumext_standar_bool
488   {
489     \IfDocumentMetadataTF
490     {
491       \__enumext_renew_footnote:
492     }
493     {
494       \bool_if:NF \l__enumext_footnotes_key_bool
495       {
496         \__enumext_renew_footnote:
497       }
498     }
499   }
500 }
501 \cs_new_protected:Nn \__enumext_print_footnote_standar:
502 {
503   \bool_if:NT \g__enumext_standar_bool
504   {
505     \IfDocumentMetadataTF
506     {
507       \__enumext_print_footnote:
508     }
509     {
510       \bool_if:NF \l__enumext_footnotes_key_bool
511       {
512         \__enumext_print_footnote:
513       }
514     }
515   }
516 }

```

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.

```

517 \cs_new_protected:Nn \__enumext_renew_footnote_starred:
518 {
519   \IfDocumentMetadataTF
520   {
521     \__enumext_renew_footnote_mini:
522   }
523   {
524     \bool_if:NF \l__enumext_footnotes_key_bool
525     {
526       \__enumext_renew_footnote_mini:
527     }
528   }

```



```

528     }
529 }
530 \cs_new_protected:Nn \__enumext_print_footnote_starred:
531 {
532     \IfDocumentMetadataTF
533     {
534         \__enumext_print_footnote_mini:
535     }
536     {
537         \bool_if:NF \__enumext_footnotes_key_bool
538         {
539             \__enumext_print_footnote_mini:
540         }
541     }
542 }

```

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

```

543 \__enumext_after_env:nn { enumext* }
544 {
545     \__enumext_print_footnote_starred:
546 }
547 \__enumext_after_env:nn { keyans* }
548 {
549     \__enumext_print_footnote_starred:
550 }

```

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

13.9 The internal minipage environment

```

\__enumext_internal_mini_page:
__enumext_mini_env*

```

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

```

551 \cs_new_protected:Nn \__enumext_internal_mini_page:
552 {
553     \int_compare:nNtT { \__enumext_level_int } = { 0 }
554     {
555         \DeclareDocumentEnvironment{\__enumext_mini_page}{m}
556         {
557             \__enumext_renew_footnote_standar:
558             \__enumext_minipage:w [ t ] { ##1 }
559             \legacy_if_gset_false:n { @minipage }
560             \skip_vertical:N \c_zero_skip
561         }
562         {
563             \skip_vertical:N \c_zero_skip
564             \__enumext_endminipage:
565             \__enumext_print_footnote_standar:
566         }
567     }
568 }

```

(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.

```

569 \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
570 \cs_new_protected:Npn \__enumext_define_counter:Nn #1 #2
571 {
572   \cs_if_exist:cTF { c@ #2 }
573   { \msg_fatal:nnn { enumext } { counters } { #2 } }
574   {
575     \tl_set:Nn #1 { #2 }
576     \newcounter { #2 }
577   }
578 }
```

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.

```
579 \__enumext_define_counter:Nn \__enumext_counter_i_tl { enumXi }
580 \__enumext_define_counter:Nn \__enumext_counter_ii_tl { enumXii }
581 \__enumext_define_counter:Nn \__enumext_counter_iii_tl { enumXiii }
582 \__enumext_define_counter:Nn \__enumext_counter_iv_tl { enumXiv }
583 \__enumext_define_counter:Nn \__enumext_counter_v_tl { enumXv }
584 \__enumext_define_counter:Nn \__enumext_counter_vi_tl { enumXvi }
585 \__enumext_define_counter:Nn \__enumext_counter_vii_tl { enumXvii }
586 \__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 \LaTeX 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.

```
587 \cs_new_protected:Npn \__enumext_register_default_label_wd:Nn #1 #2
588 {
589   \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
590   \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
591 }
592 \__enumext_register_default_label_wd:Nn \arabic { 0 }
593 \__enumext_register_default_label_wd:Nn \Alph { M }
594 \__enumext_register_default_label_wd:Nn \alph { m }
595 \__enumext_register_default_label_wd:Nn \Roman { VIII }
596 \__enumext_register_default_label_wd:Nn \roman { viii }
```

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

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

```
597 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
598 {
599   \hbox_set:Nn \__enumext_label_width_by_box {#2}
600   \dim_set:Nn #1 { \box_wd:N \__enumext_label_width_by_box }
601 }
602 \cs_generate_variant:Nn \__enumext_label_width_by_box:Nn { cv }
```

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

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.

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

```

604 {
605   \tl_clear_new:N #1
606   \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
607   \tl_gset_eq:NN \g__enumext_widest_label_tl #1
608   \tl_map_inline:Nn \g__enumext_counter_styles_tl
609     {
610       \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
611       \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
612       { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
613     }
614   \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
615   { \tl_use:N \g__enumext_widest_label_tl }
616   \tl_set_eq:cN { the #2 } #1
617 }
618 \cs_generate_variant:Nn \__enumext_label_style:Nnn { cvn }

```

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

13.13 Setting keys associated with label

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

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

```

619 \cs_set_protected:Npn \__enumext_tmp:n #1
620 {
621   \keys_define:nn { enumext / #1 }
622   {
623     mode-box .bool_set:N = \l__enumext_mode_box_bool,
624     mode-box .initial:n = false,
625     mode-box .value_forbidden:n = true,
626   }
627 }
628 \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.

```

629 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
630 {
631   \keys_define:nn { enumext / #1 }
632   {
633     font .tl_set:c = { \__enumext_label_font_style_#2_tl },
634     font .value_required:n = true,
635     labelsep .dim_set:c = { \__enumext_labelsep_#2_dim },
636     labelsep .initial:n = {0.3333em},
637     labelsep .value_required:n = true,
638     labelwidth .dim_set:c = { \__enumext_labelwidth_#2_dim },
639     labelwidth .value_required:n = true,
640     wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
641     wrap-label .initial:n = {##1},
642     wrap-label .value_required:n = true,
643     wrap-label* .code:n = {
644       \bool_set_true:c { \__enumext_wrap_label_opt_#2_bool }
645       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
646     },
647     wrap-label* .value_required:n = true,
648   }
649 }
650 \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`.

```

651 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
652 {
653   \keys_define:nn { enumext / #1 }
654   {
655     align .choice:,

```

```

656         align / left      .code:n =
657                             {
658                                 \tl_clear:c { l__enumext_label_fill_left_#2_tl }
659                                 \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
660                                 \str_set:cn { l__enumext_align_label_pos_#2_str } { l }
661                             },
662         align / right     .code:n =
663                             {
664                                 \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
665                                 \tl_clear:c { l__enumext_label_fill_right_#2_tl }
666                                 \str_set:cn { l__enumext_align_label_pos_#2_str } { r }
667                             },
668         align / center    .code:n =
669                             {
670                                 \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
671                                 \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
672                                 \str_set:cn { l__enumext_align_label_pos_#2_str } { c }
673                             },
674         align / unknown   .code:n =
675                             \msg_error:nneee { enumext } { unknown-choice }
676                             { align } { left,~right,~ center } { \exp_not:n {##1} },
677         align .initial:n = left,
678         align .value_required:n = true,
679     }
680 }
681 \clist_map_inline:nn
682 {
683     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
684 }
685 { \__enumext_tmp:nn #1 }

686 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
687 {
688     \keys_define:nn { enumext / #1 }
689     {
690         align .choice:,
691         align / left      .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
692         align / right     .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
693         align / center    .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
694         align / unknown   .code:n =
695                             \msg_error:nneee { enumext } { unknown-choice }
696                             { align } { left,~right,~ center } { \exp_not:n {##1} },
697         align .initial:n = left,
698         align .value_required:n = true,
699     }
700 }
701 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

(End of definition for align.)

13.14 Setting label and ref keys

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

13.14.1 Define and set label and ref keys for enumext environment

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

```

\l__enumext_label_i_tl
\l__enumext_label_ii_tl
\l__enumext_label_iii_tl
\l__enumext_label_iv_tl

702 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
703 {
704     \keys_define:nn { enumext / #1 }
705     {
706         label .code:n = {
707             \__enumext_label_style:cvn { l__enumext_label_#2_tl }
708             { l__enumext_counter_#2_tl } {##1}
709             \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
710             \l__enumext_current_widest_dim
711         },
712         label .initial:n = #3,
713         label .value_required:n = true,
714         ref .code:n = \__enumext_standar_ref:n {##1},

```

```

715         ref .value_required:n = true,
716     }
717 }
718 \__enumext_tmp:nnn { level-1 } { i } { \arabic*.}
719 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*) }
720 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
721 \__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }

```

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

```

\__enumext_standar_ref:n
\__enumext_standar_ref:

```

The `__enumext_standard_ref:n` function will first pass the key *argument* `ref` to the variable `\l__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 `\l__enumext_the_counter_X_tl` to `\l__enumext_ref_the_count_tl` and then set the variable `\l__enumext_renew_counter_X_tl` which will modify `\theenumX`.

```

722 \cs_new_protected:Npn \__enumext_standar_ref:n #1
723 {
724     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
725     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
726     {
727         \msg_error:nnn { enumext } { key-ref-empty } { enumext }
728     }
729     {
730         \tl_set_eq:Nc \l__enumext_ref_the_count_tl
731         {
732             l__enumext_the_counter_ \__enumext_level: _tl
733         }
734         \tl_set:ce { l__enumext_renew_counter_ \__enumext_level: _tl }
735         {
736             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
737             { \exp_not:V \l__enumext_ref_key_arg_tl }
738         }
739     }
740 }

```

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

```

741 \cs_new_protected:Npn \__enumext_standar_ref:
742 {
743     \tl_if_empty:cF { l__enumext_renew_counter_ \__enumext_level: _tl }
744     {
745         \tl_use:c { l__enumext_renew_counter_ \__enumext_level: _tl }
746     }
747 }

```

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

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

```

label
ref

```

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

```

\l__enumext_label_vii_tl
\l__enumext_label_viii_tl
748 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
749 {
750     \keys_define:nn { enumext / #1 }
751     {
752         label .code:n = {
753             \__enumext_label_style:cvn { l__enumext_label_#2_tl }
754             { l__enumext_counter_#2_tl } {##1}
755             \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
756             \l__enumext_current_widest_dim
757         },
758         label .initial:n = #3,
759         label .value_required:n = true,
760         ref .code:n = \__enumext_starred_ref:n {##1},
761         ref .value_required:n = true,
762     }
763 }
764 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
765 \__enumext_tmp:nnn { keyans* } { viii } { \Alph*) }

```

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

`__enumext_starred_ref:n`
`__enumext_starred_ref:`

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

```

766 \cs_new_protected:Npn \__enumext_starred_ref:n #1
767 {
768   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
769   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
770   {
771     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
772     {
773       \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
774     }
775     {
776       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
777       \tl_set:Ne \l__enumext_renew_counter_vii_tl
778       {
779         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V
780       }
781     }
782   }
783   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
784   {
785     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
786     {
787       \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
788     }
789     {
790       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
791       \tl_set:Ne \l__enumext_renew_counter_viii_tl
792       {
793         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V
794       }
795     }
796   }
797 }

```

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.

```

798 \cs_new_protected:Nn \__enumext_starred_ref:
799 {
800   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
801   {
802     \tl_if_empty:NF \l__enumext_renew_counter_vii_tl
803     {
804       \tl_use:N \l__enumext_renew_counter_vii_tl
805     }
806   }
807   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
808   {
809     \tl_if_empty:NF \l__enumext_renew_counter_viii_tl
810     {
811       \tl_use:N \l__enumext_renew_counter_viii_tl
812     }
813   }
814 }

```

(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.

```

\l__enumext_label_v_tl
\l__enumext_label_vi_tl
815 \keys_define:nn { enumext / keyans }
816 {
817   label .code:n = {
818     \__enumext_label_style:cvn { \l__enumext_label_v_tl }
819     { \l__enumext_counter_v_tl } {#1}
820     \__enumext_label_style:cvn { \l__enumext_label_vi_tl }
821     { \l__enumext_counter_vi_tl } {#1}
822     \dim_set_eq:NN
823     \l__enumext_labelwidth_v_dim \l__enumext_current_widest_dim
824   },
825   label .initial:n = \Alph*,

```



```

826     label .value_required:n = true,
827     ref .code:n = \__enumext_keyans_ref:n {#1},
828     ref .value_required:n = true,
829 }

```

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

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

```

830 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
831 {
832   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
833   \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
834   {
835     \msg_error:nnn { enumext } { key-ref-empty } { keyans }
836   }
837   {
838     \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
839     \tl_put_right:Ne \l__enumext_renew_counter_v_tl
840     {
841       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V \l__
842     }
843   }
844 }

```

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

```

845 \cs_new_protected:Nn \__enumext_keyans_ref:
846 {
847   \tl_if_empty:NF \l__enumext_renew_counter_v_tl
848   {
849     \tl_use:N \l__enumext_renew_counter_v_tl
850   }
851 }

```

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

13.15 Setting `start`, `start*` and widest keys

`__enumext_start_from:NNn` 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`
 #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.

```

852 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
853 {
854   \__enumext_if_is_int:nTF { #3 }
855   {
856     \int_set:Nn #2 {#3}
857   }
858   {
859     \regex_if_match:nVT { \c{Alph} | \c{alph} } {#1}
860     { \int_set:Nn #2 { \int_from_alph:n {#3} } }
861     \regex_if_match:nVT { \c{Roman} | \c{roman} } {#1}
862     { \int_set:Nn #2 { \int_from_roman:n {#3} } }
863   }
864 }
865 \cs_generate_variant:Nn \__enumext_start_from:NNn { ccn, cce }

```

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

`__enumext_widest_from:nNNn` The function `__enumext_widest_from:nNNn` used by the `widest` key take four arguments:
`__enumext_widest_from:nccn` #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 fourth argument can be an *integer* or *string* of the form `\Alph`, `\alph`, `\Roman` or `\roman`. The value of the fourth 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.

```

866 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
867 {
868   \__enumext_if_is_int:nTF {#4}
869   {
870     \setcounter{enumX#1} { #4 }
871   }
872   {
873     \regex_if_match:nVT { \c{Alph} | \c{alph} } {#2}
874     { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
875     \regex_if_match:nVT { \c{Roman} | \c{roman} } {#2}
876     { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
877   }
878   \__enumext_label_width_by_box:cv
879   { l__enumext_labelwidth_#1_dim } { l__enumext_label_#1_tl }
880 }
881 \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.

```

882 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
883 {
884   \keys_define:nn { enumext / #1 }
885   {
886     start* .code:n = {
887       \__enumext_start_from:ccn
888       { l__enumext_label_#2_tl }
889       { l__enumext_start_#2_int } {##1}
890     },
891     start* .value_required:n = true,
892     start .code:n = {
893       \__enumext_start_from:cce
894       { l__enumext_label_#2_tl }
895       { l__enumext_start_#2_int } { \int_eval:n {##1} }
896     },
897     start .initial:n = 1,
898     start .value_required:n = true,
899     widest .code:n = {
900       \__enumext_widest_from:nccn {#2}
901       { l__enumext_label_#2_tl }
902       { l__enumext_labelwidth_#2_dim } {##1}
903     },
904     widest .value_required:n = true,
905   }
906 }
907 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

13.16 Setting keys for vertical spaces

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

```

908 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
909 {
910   \keys_define:nn { enumext / #1 }
911   {
912     topsep .skip_set:c = { l__enumext_topsep_#2_skip },
913     topsep .initial:n = {#3},
914     topsep .value_required:n = true,
915     partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
916     partopsep .initial:n = {#4},
917     partopsep .value_required:n = true,
918     parsep .skip_set:c = { l__enumext_parsep_#2_skip },
919     parsep .initial:n = {#5},
920     parsep .value_required:n = true,

```

```

921     itemsep .skip_set:c = { l__enumext_itemsep_#2_skip },
922     itemsep .initial:n = {#6},
923     itemsep .value_required:n = true,
924     noitemsep .meta:n = { itemsep = 0pt, parsep = 0pt },
925     noitemsep .value_forbidden:n = true,
926     nosepe .meta:n = {
927         itemsep = 0pt, parsep= 0pt,
928         topsep = 0pt, partopsep = 0pt,
929     },
930     nosepe .value_forbidden:n = true,
931 }
932 }

```

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

```

933 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
934 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
935 { 4.0pt plus 2.0pt minus 1.0pt }
936 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
937 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
938 { 2.0pt plus 1.0pt minus 1.0pt }
939 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
940 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
941 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
942 { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
943 \__enumext_tmp:nnnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
944 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
945 { 2.0pt plus 1.0pt minus 1.0pt }
946 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
947 { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
948 { 4.0pt plus 2.0pt minus 1.0pt }
949 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
950 { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
951 { 2.0pt plus 1.0pt minus 1.0pt }

```

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

13.17 Setting base-fix key

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

```

base-fix \__enumext_nested_base_line_fix:
952 \keys_define:nn { enumext / level-1 }
953 {
954     base-fix .bool_set:N = \l__enumext_base_line_fix_bool,
955     base-fix .initial:n = false,
956     base-fix .value_forbidden:n = true,
957 }

```

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

We will first implement the function code from the user side of the `base-fix` key, that is, only the user knows when it is necessary to apply it within the document in which case the variable `\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.

```

958 \cs_new_protected:Nn \__enumext_nested_base_line_fix:
959 {
960     \bool_lazy_all:nT
961     {
962         { \bool_if_p:N \l__enumext_starred_first_bool }
963         { \bool_if_p:N \l__enumext_base_line_fix_bool }
964         { \bool_not_p:n { \l__enumext_print_keyans_star_bool } }
965     }
966     {
967         \mode_leave_vertical:

```

```

968     \vspace { -\dim_eval:n { \baselineskip + \parsep } }
969     \keys_set:nn { enumext / level-1 }
970     {
971         topsep = 0pt, above = 0pt, above* = 0pt,
972     }
973 }

```

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`.

```

974     \bool_lazy_and:nnT
975     { \bool_if_p:N \l__enumext_starred_first_bool }
976     { \bool_if_p:N \l__enumext_print_keyans_star_bool }
977     {
978         \mode_leave_vertical:
979         \skip_vertical:n { -\baselineskip }
980         \skip_vertical:N \c_zero_skip
981         \keys_set:nn { enumext / level-1 }
982         {
983             topsep = 0pt, above = 0pt, above* = 0pt,
984         }
985     }
986     \bool_set_false:N \l__enumext_base_line_fix_bool
987 }

```

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

13.18 Setting keys for horizontal spaces

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

```

988 \cs_set_protected:Npn \l__enumext_tmp:nn #1 #2
989 {
990     \keys_define:nn { enumext / #1 }
991     {
992         itemindent .dim_set:c = { \l__enumext_fake_item_indent_#2_dim },
993         itemindent .value_required:n = true,
994         rightmargin .dim_set:c = { \l__enumext_rightmargin_#2_dim },
995         rightmargin .value_required:n = true,
996         listparindent .dim_set:c = { \l__enumext_listparindent_#2_dim },
997         listparindent .value_required:n = true,
998         list-offset .dim_set:c = { \l__enumext_listoffset_#2_dim },
999         list-offset .value_required:n = true,
1000         list-indent .code:n =
1001             \bool_set_true:c { \l__enumext_leftmargin_tmp_#2_bool }
1002             \dim_set:cn { \l__enumext_leftmargin_tmp_#2_dim } {##1},
1003         list-indent .value_required:n = true,
1004     }
1005 }
1006 \clist_map_inline:nn
1007 {
1008     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
1009 }
1010 { \l__enumext_tmp:nn #1 }

```

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

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

```

1011 \cs_set_protected:Npn \l__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 .meta:n = { list-offset = ##1 },
1024         list-indent .value_required:n = true,

```

```

1025     }
1026   }
1027   \clist_map_inline:nn
1028   {
1029     {enumext*}{vii}, {keyans*}{viii}
1030   }
1031   { \__enumext_tmp:nn #1 }

```

13.18.1 Functions for setting the fake itemindent

The `itemindent` key does not set the value of `\itemindent`, it only sets the value of the *horizontal space* applied using `\skip_horizontal:N`. We will store this value in the variable and only apply it when it is greater than `\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.

```

1032 \cs_set_protected:Nn \__enumext_fake_item_indent:
1033 {
1034   \dim_compare:nNnT
1035   { \dim_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _dim } }
1036   >
1037   { \c_zero_dim }
1038   {
1039     \tl_set:ce { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
1040     {
1041       \exp_not:N \mode_leave_vertical:
1042       \exp_not:n { \skip_horizontal:n }
1043       { \dim_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _dim } }
1044       \exp_not:N \ignorespaces
1045     }
1046   }
1047 }
1048 \cs_set_protected:Nn \__enumext_keyans_fake_item_indent:
1049 {
1050   \dim_compare:nNnT
1051   { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
1052   {
1053     \tl_set:Ne \l__enumext_fake_item_indent_v_tl
1054     {
1055       \exp_not:N \mode_leave_vertical:
1056       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
1057       \exp_not:N \ignorespaces
1058     }
1059   }
1060 }
1061 \cs_set_protected:Nn \__enumext_fake_item_indent_vii:
1062 {
1063   \dim_compare:nNnT
1064   { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
1065   {
1066     \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
1067     {
1068       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
1069       \exp_not:N \ignorespaces
1070     }
1071   }
1072 }
1073 \cs_set_protected:Nn \__enumext_fake_item_indent_viii:
1074 {
1075   \dim_compare:nNnT
1076   { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
1077   {
1078     \tl_set:Ne \l__enumext_fake_item_indent_viii_tl
1079     {
1080       \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
1081       \exp_not:N \ignorespaces
1082     }
1083   }
1084 }

```

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

13.19 Setting show-length key

show-length

Define and set `show-length` key for `enumext`, `enumext*`, `keyans` and `keyans*` environments. The function sets the boolean variable `__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.

```

1085 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1086 {
1087   \keys_define:nn { enumext / #1 }
1088   {
1089     show-length .bool_set:c = { l__enumext_show_length_#2_bool },
1090     show-length .initial:n = false,
1091   }
1092 }
1093 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

13.20 Setting before, after and first keys

before

before*

after

first

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

```

1094 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1095 {
1096   \keys_define:nn { enumext / #1 }
1097   {
1098     before .tl_set:c = { l__enumext_before_no_starred_key_#2_tl },
1099     before .value_required:n = true,
1100     before* .tl_set:c = { l__enumext_before_starred_key_#2_tl },
1101     before* .value_required:n = true,
1102     after .tl_set:c = { l__enumext_after_stop_list_#2_tl },
1103     after .value_required:n = true,
1104     first .tl_set:c = { l__enumext_after_list_args_#2_tl },
1105     first .value_required:n = true,
1106   }
1107 }
1108 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

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

13.20.1 Functions for before, after and first keys in enumext

__enumext_before_args_exec:

__enumext_before_keys_exec:

__enumext_after_stop_list:

__enumext_after_args_exec:

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

```

1109 \cs_new_protected:Nn \__enumext_before_args_exec:
1110 {
1111   \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
1112 }

```

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⟩}`

```

1113 \cs_new_protected:Nn \__enumext_before_keys_exec:
1114 {
1115   \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
1116 }

```

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

```

1117 \cs_new_protected:Nn \__enumext_after_stop_list:
1118 {
1119   \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
1120 }

```

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.`

```

1121 \cs_new_protected:Nn \__enumext_after_args_exec:
1122 {
1123   \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
1124 }

```

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

13.20.2 Functions for before, after and first keys in keyans

Same implementation as the one used in the [enumext](#) environment.

```

__enumext_before_args_exec_v:
__enumext_before_keys_exec_v:
__enumext_after_stop_list_v:
__enumext_after_args_exec_v:
1125 \cs_new_protected:Nn __enumext_before_args_exec_v:
1126 {
1127   \tl_use:N \l__enumext_before_starred_key_v_tl
1128 }
1129 \cs_new_protected:Nn __enumext_before_keys_exec_v:
1130 {
1131   \tl_use:N \l__enumext_before_no_starred_key_v_tl
1132 }
1133 \cs_new_protected:Nn __enumext_after_stop_list_v:
1134 {
1135   \tl_use:N \l__enumext_after_stop_list_v_tl
1136 }
1137 \cs_new_protected:Nn __enumext_after_args_exec_v:
1138 {
1139   \tl_use:N \l__enumext_after_list_args_v_tl
1140 }

```

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

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

Same implementation as the one used in the [enumext](#) environment.

```

__enumext_before_args_exec_vii:
__enumext_before_keys_exec_vii
__enumext_after_stop_list_vii:
__enumext_after_args_exec_vii:
1141 \cs_new_protected:Nn __enumext_before_args_exec_vii:
1142 {
1143   \tl_use:N \l__enumext_before_starred_key_vii_tl
1144 }
1145 \cs_new_protected:Nn __enumext_before_args_exec_viii:
1146 {
1147   \tl_use:N \l__enumext_before_starred_key_viii_tl
1148 }
1149 \cs_new_protected:Nn __enumext_before_keys_exec_vii:
1150 {
1151   \tl_use:N \l__enumext_before_no_starred_key_vii_tl
1152 }
1153 \cs_new_protected:Nn __enumext_before_keys_exec_viii:
1154 {
1155   \tl_use:N \l__enumext_before_no_starred_key_viii_tl
1156 }
1157 \cs_new_protected:Nn __enumext_after_stop_list_vii:
1158 {
1159   \tl_use:N \l__enumext_after_stop_list_vii_tl
1160 }
1161 \cs_new_protected:Nn __enumext_after_stop_list_viii:
1162 {
1163   \tl_use:N \l__enumext_after_stop_list_viii_tl
1164 }
1165 \cs_new_protected:Nn __enumext_after_args_exec_vii:
1166 {
1167   \tl_use:N \l__enumext_after_list_args_vii_tl
1168 }
1169 \cs_new_protected:Nn __enumext_after_args_exec_viii:
1170 {
1171   \tl_use:N \l__enumext_after_list_args_viii_tl
1172 }

```

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

13.21 Setting keys for multicols and minipage

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

```

1173 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1174 {
1175   \keys_define:nn { enumext / #1 }
1176   {
1177     mini-env .dim_set:c = { l__enumext_minipage_right_#2_dim },
1178     mini-env .value_required:n = true,
1179     mini-sep .dim_set:c = { l__enumext_minipage_hsep_#2_dim },

```

```

1180     mini-sep .initial:n = 0.3333em,
1181     mini-sep .value_required:n = true,
1182     columns-sep .dim_set:c = { l__enumext_columns_sep_#2_dim },
1183     columns-sep .value_required:n = true,
1184     columns .int_set:c = { l__enumext_columns_#2_int },
1185     columns .initial:n = 1,
1186     columns .value_required:n = true,
1187   }
1188 }
1189 \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.

```

1190 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1191 {
1192   \keys_define:nn { enumext / #1 }
1193   {
1194     mini-right .tl_gset:c = { g__enumext_miniright_code_#2_tl },
1195     mini-right .value_required:n = true,
1196     mini-right* .code:n = {
1197       \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
1198       \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1199     },
1200     mini-right* .value_required:n = true,
1201   }
1202 }
1203 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

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

13.22 Adjustment of vertical spaces for multicols

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

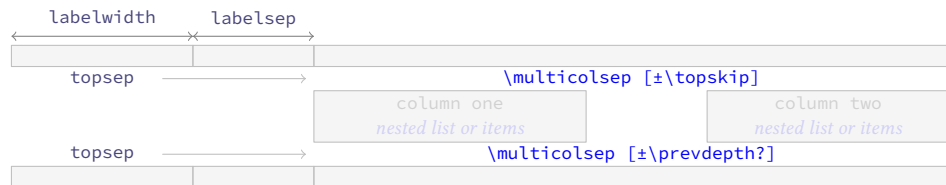


Figure 7: Representation of the vertical space in `multicols` for a nested level.

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

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

13.22.1 Adjustment of vertical spaces for multicols in enumext

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

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

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

```

1204 \cs_new_protected:Nn \__enumext_multi_set_vskip:
1205 {
1206   \skip_set:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1207   {
1208     \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1209   }
1210   \skip_set:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1211   {
1212     \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1213   }

```

```

1214   \__enumext_add_pre_parsep:
1215   }

```

(End of definition for __enumext_multi_set_vskip:.)

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

```

1216 \cs_new_protected:Nn \__enumext_add_pre_parsep:
1217 {
1218   \int_case:nn { \l__enumext_level_int }
1219   {
1220     { 2 }{
1221       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1222       {
1223         \skip_add:Nn \l__enumext_multicols_above_ii_skip
1224         {
1225           \l__enumext_parsep_i_skip
1226         }
1227       }
1228     }
1229     { 3 }{
1230       \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1231       {
1232         \skip_add:Nn \l__enumext_multicols_above_iii_skip
1233         {
1234           \l__enumext_parsep_ii_skip
1235         }
1236       }
1237     }
1238     { 4 }{
1239       \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1240       {
1241         \skip_add:Nn \l__enumext_multicols_above_iv_skip
1242         {
1243           \l__enumext_parsep_iii_skip
1244         }
1245       }
1246     }
1247   }
1248 }

```

(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⟩.

```

1249 \cs_new_protected:Nn \__enumext_multi_addvspace:
1250 {
1251   \__enumext_multi_set_vskip:
1252   \mode_if_vertical:T
1253   {
1254     \skip_add:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1255     {
1256       \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1257     }
1258     \skip_add:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1259     {
1260       \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1261     }
1262   }
1263   \par\nopagebreak
1264   \addvspace{ \skip_use:c { l__enumext_multicols_above_ \__enumext_level: _skip } }
1265 }

```

(End of definition for __enumext_multi_addvspace:.)

13.22.2 Adjustment of vertical spaces for multicol in keyans

`__enumext_keyans_multi_set_vskip:`
`__enumext_keyans_multi_addvspace:`

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

```

1266 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1267 {
1268   \skip_set:Nn \l__enumext_multicol_above_v_skip
1269   {
1270     \l__enumext_topsep_v_skip
1271   }
1272   \skip_set:Nn \l__enumext_multicol_below_v_skip
1273   {
1274     \l__enumext_topsep_v_skip
1275   }
1276 }
1277 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1278 {
1279   \__enumext_keyans_multi_set_vskip:
1280   \mode_if_vertical:T
1281   {
1282     \skip_add:Nn \l__enumext_multicol_above_v_skip
1283     {
1284       \skip_use:N \l__enumext_partopsep_v_skip
1285     }
1286     \skip_add:Nn \l__enumext_multicol_below_v_skip
1287     {
1288       \skip_use:N \l__enumext_partopsep_v_skip
1289     }
1290   }
1291   \par\nopagebreak
1292   \addvspace{ \l__enumext_multicol_above_v_skip }
1293 }

```

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

13.23 Adjustment of vertical spaces for minipage

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



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

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

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

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

13.23.1 Adjustment of vertical spaces for minipage in enumext

`__enumext_minipage_set_skip:`
`__enumext_minipage_add_space:`

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

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

```

1294 \cs_new_protected:Nn \__enumext_minipage_set_skip:
1295 {
1296   \skip_set:Nn \l__enumext_minipage_right_skip

```

```

1297     {
1298       \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1299     }
1300     \mode_if_vertical:T
1301     {
1302       \skip_add:Nn \l__enumext_minipage_right_skip
1303       {
1304         \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1305       }
1306     }
1307     \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:`.

```

1308     \skip_set_eq:cN
1309     { l__enumext_multicols_above_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1310     \skip_set_eq:cN
1311     { l__enumext_multicols_below_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1312     \__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`.

```

1313     \int_compare:nNnT
1314     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1315     {
1316       \skip_zero:N \topskip
1317       \skip_set_eq:Nc \multicolsep { l__enumext_multicols_above_ \__enumext_level: _skip }
1318     }
1319   }

```

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

```

1320 \cs_new_protected:Nn \__enumext_minipage_add_space:
1321 {
1322   \__enumext_minipage_set_skip:
1323   \__enumext_unskip_unkern:
1324   \mode_if_vertical:TF
1325   {
1326     \nopagebreak\nointerlineskip
1327   }
1328   {
1329     \par\nopagebreak\nointerlineskip
1330     \skip_zero:c { l__enumext_partopsep_ \__enumext_level: _skip }
1331   }
1332   \int_compare:nNnTF
1333   { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1334   {
1335     \addvspace{ 0.445\box_ht:N \strutbox }
1336   }
1337   {
1338     \addvspace{ 0.250\box_ht:N \strutbox }
1339   }
1340 }

```

(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.

```

1341 \cs_new_protected:Nn \__enumext_pre_itemsep_skip:
1342 {
1343   \int_case:nn { \l__enumext_level_int }
1344   {
1345     { 2 }{
1346       \skip_if_eq:nnTF
1347       { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1348       {
1349         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1350         \skip_set:Nn \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }

```

```

1351     }
1352     {
1353         \dim_compare:nNnT
1354         { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1355         {
1356             \skip_sub:Nn
1357             \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1358             \skip_sub:Nn
1359             \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1360             \skip_add:Nn
1361             \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1362             \skip_add:Nn
1363             \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1364         }
1365         \dim_compare:nNnT
1366         { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1367         {
1368             \skip_set:Nn \l__enumext_minipage_temp_skip
1369             {
1370                 \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1371             }
1372             \skip_sub:Nn
1373             \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1374             \skip_sub:Nn
1375             \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1376             \skip_add:Nn
1377             \l__enumext_minipage_after_skip
1378             { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1379             \skip_add:Nn
1380             \l__enumext_multicols_below_ii_skip
1381             { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1382         }
1383     }
1384 }
1385 { 3 }{
1386     \skip_if_eq:nnTF
1387     { \l__enumext_itemsep_ii_skip } { \c_zero_skip }
1388     {
1389         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1390         \skip_set:Nn \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1391     }
1392     {
1393         \dim_compare:nNnT
1394         { \l__enumext_itemsep_ii_skip } < { \l__enumext_minipage_after_skip }
1395         {
1396             \skip_sub:Nn
1397             \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1398             \skip_sub:Nn
1399             \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1400             \skip_add:Nn
1401             \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1402             \skip_add:Nn
1403             \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1404         }
1405         \dim_compare:nNnT
1406         { \l__enumext_itemsep_ii_skip } > { \l__enumext_minipage_after_skip }
1407         {
1408             \skip_set:Nn \l__enumext_minipage_temp_skip
1409             {
1410                 \l__enumext_itemsep_ii_skip - \l__enumext_minipage_after_skip
1411             }
1412             \skip_sub:Nn
1413             \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1414             \skip_sub:Nn
1415             \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1416             \skip_add:Nn
1417             \l__enumext_minipage_after_skip
1418             { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1419             \skip_add:Nn
1420             \l__enumext_multicols_below_iii_skip
1421             { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }

```



```

1422         }
1423     }
1424 }
1425 { 4 }{
1426     \skip_if_eq:nnTF { \l__enumext_itemsep_iii_skip } { \c_zero_skip }
1427     {
1428         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1429         \skip_set:Nn \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1430     }
1431     {
1432         \dim_compare:nNnT
1433         { \l__enumext_itemsep_iii_skip } < { \l__enumext_minipage_after_skip }
1434         {
1435             \skip_sub:Nn
1436             \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1437             \skip_sub:Nn
1438             \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1439             \skip_add:Nn
1440             \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1441             \skip_add:Nn
1442             \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1443         }
1444         \dim_compare:nNnT
1445         { \l__enumext_itemsep_iii_skip } > { \l__enumext_minipage_after_skip }
1446         {
1447             \skip_set:Nn \l__enumext_minipage_temp_skip
1448             {
1449                 \l__enumext_itemsep_iii_skip - \l__enumext_minipage_after_skip
1450             }
1451             \skip_sub:Nn
1452             \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1453             \skip_sub:Nn
1454             \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1455             \skip_add:Nn
1456             \l__enumext_minipage_after_skip
1457             { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1458             \skip_add:Nn
1459             \l__enumext_multicols_below_iv_skip
1460             { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1461         }
1462     }
1463 }
1464 }
1465 }

```

(End of definition for `__enumext_pre_itemsep_skip`.)

13.23.2 Adjustment of vertical spaces for minipage in keyans

```

\__enumext_keyans_minipage_set_skip:
\__enumext_keyans_minipage_add_space:
\__enumext_keyans_pre_itemsep_skip:

```

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

```

1466 \cs_new_protected:Nn \__enumext_keyans_minipage_set_skip:
1467 {
1468     \skip_zero:N \l__enumext_minipage_after_skip
1469     \skip_zero:N \l__enumext_minipage_left_skip
1470     \skip_zero:N \l__enumext_minipage_right_skip
1471     \skip_set:Nn \l__enumext_minipage_right_skip
1472     {
1473         \l__enumext_topsep_v_skip
1474     }
1475     \mode_if_vertical:T
1476     {
1477         \skip_add:Nn \l__enumext_minipage_right_skip
1478         {
1479             \l__enumext_partopsep_v_skip
1480         }
1481     }
1482     \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip
1483     \skip_set_eq:NN \l__enumext_multicols_above_v_skip \l__enumext_minipage_right_skip
1484     \skip_set_eq:NN \l__enumext_multicols_below_v_skip \l__enumext_minipage_right_skip
1485     \__enumext_keyans_pre_itemsep_skip:

```

```

1486 \int_compare:nNt { \l__enumext_columns_v_int } > { 1 }
1487 {
1488   \skip_zero:N \topskip
1489   \skip_set_eq:NN \multicolsep \l__enumext_minipage_right_skip
1490 }
1491 }
1492 \cs_new_protected:Nn \l__enumext_keyans_minipage_add_space:
1493 {
1494   \l__enumext_keyans_minipage_set_skip:
1495   \l__enumext_unskip_unkern:
1496   \mode_if_vertical:TF
1497   {
1498     \nopagebreak\nointerlineskip
1499   }
1500   {
1501     \par\nopagebreak\nointerlineskip
1502     \skip_zero:N \l__enumext_partopsep_v_skip
1503   }
1504   \int_compare:nNtTF { \l__enumext_columns_v_int } > { 1 }
1505   {
1506     \addvspace{ 0.445\box_ht:N \strutbox }
1507   }
1508   {
1509     \addvspace{ 0.250\box_ht:N \strutbox }
1510   }
1511 }
1512 \cs_new_protected:Nn \l__enumext_keyans_pre_itemsep_skip:
1513 {
1514   \skip_if_eq:nnTF
1515   { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1516   {
1517     \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1518     \skip_set:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1519   }
1520   {
1521     \dim_compare:nNt
1522     { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1523     {
1524       \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1525       \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1526       \skip_add:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1527       \skip_add:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1528     }
1529     \dim_compare:nNt
1530     { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1531     {
1532       \skip_set:Nn \l__enumext_minipage_temp_skip
1533       {
1534         \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1535       }
1536       \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1537       \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1538       \skip_add:Nn \l__enumext_minipage_after_skip
1539       { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1540       \skip_add:Nn \l__enumext_multicols_below_v_skip
1541       { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1542     }
1543   }
1544 }

```

(End of definition for `\l__enumext_keyans_minipage_set_skip:`, `\l__enumext_keyans_minipage_add_space:`, and `\l__enumext_keyans_pre_itemsep_skip:`.)

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

`\l__enumext_mini_set_vskip_vii:`
`\l__enumext_mini_set_vskip_viii:`

The functions `\l__enumext_mini_set_vskip_vii:` and `\l__enumext_mini_set_vskip_viii:` will take care of determining the “adjusted” spaces that we will apply “above” and “below” the `\l__enumext_mini_page` environment in `enumext*` and `keyans*`.

```

1545 \cs_new_protected:Nn \l__enumext_mini_set_vskip_vii:
1546 {
1547   \skip_zero_new:N \l__enumext_minipage_left_skip
1548   \skip_gzero_new:N \l__enumext_minipage_right_skip

```

```

1549 \skip_gzero_new:N \g__enumext_minipage_after_skip
1550 \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1551 {
1552   \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1553   \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1554 }
1555 {
1556   \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1557   \skip_gset:Nn \g__enumext_minipage_right_skip
1558   {
1559     \l__enumext_topsep_vii_skip
1560   }
1561   \skip_gset:Nn \g__enumext_minipage_after_skip
1562   {
1563     0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1564   }
1565 }
1566 }
1567 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1568 {
1569   \skip_zero_new:N \l__enumext_minipage_after_skip
1570   \skip_zero_new:N \l__enumext_minipage_left_skip
1571   \skip_zero_new:N \l__enumext_minipage_right_skip
1572   \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1573   {
1574     \skip_set:Nn \l__enumext_minipage_left_skip
1575     {
1576       0.5\box_dp:N \strutbox
1577     }
1578     \skip_set:Nn \l__enumext_minipage_right_skip
1579     {
1580       \l__enumext_partopsep_viii_skip
1581     }
1582     \skip_set:Nn \l__enumext_minipage_after_skip
1583     {
1584       1.6\box_dp:N \strutbox
1585     }
1586   }
1587   {
1588     \skip_set:Nn \l__enumext_minipage_left_skip
1589     {
1590       0.5875\box_dp:N \strutbox
1591     }
1592     \skip_set:Nn \l__enumext_minipage_right_skip
1593     {
1594       \l__enumext_topsep_viii_skip
1595     }
1596     \skip_set:Nn \l__enumext_minipage_after_skip
1597     {
1598       0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1599     }
1600   }
1601 }

```

(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.

```

1602 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1603 {
1604   \__enumext_mini_set_vskip_vii:
1605   \par\nopagebreak
1606   \addvspace { \l__enumext_minipage_left_skip }
1607 }
1608 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1609 {
1610   \__enumext_mini_set_vskip_viii:
1611   \par\nopagebreak

```

```

1612     \addvspace { \l__enumext_minipage_left_skip }
1613   }

```

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

13.23.4 The command `\miniright`

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

```

1614 \NewDocumentCommand \miniright { s }
1615 {
1616   \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
1617   {
1618     \msg_error:nnn { enumext } { wrong-miniright-place }
1619   }
1620   % outside
1621   \bool_lazy_and:nnT
1622   { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
1623   { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
1624   {
1625     \msg_error:nnn { enumext } { wrong-miniright-place }
1626   }
1627   % starred env
1628   \bool_lazy_and:nnT
1629   { \bool_if_p:N \g__enumext_starred_bool }
1630   { \bool_not_p:n { \l__enumext_standar_bool } }
1631   {
1632     \msg_error:nnn { enumext } { wrong-miniright-starred }
1633   }
1634   % exec
1635   \int_compare:nNtF { \l__enumext_keyans_level_int } = { 1 }
1636   {
1637     \__enumext_keyans_mini_right_cmd:n {#1}
1638   }
1639   { \__enumext_mini_right_cmd:n {#1} }
1640 }

```

(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.

```

1641 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1642 {
1643   \dim_compare:nNtF
1644   { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1645   {
1646     \__enumext_multicols_stop:
1647     \int_compare:nNtF
1648     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } = { 1 }
1649     {
1650       \par\addvspace{ \l__enumext_minipage_after_skip }
1651     }
1652     \end__enumext_mini_page
1653     \hfill
1654     \__enumext_mini_page{ \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } }
1655     \par\nointerlineskip
1656     \addvspace { \l__enumext_minipage_right_skip }
1657     \bool_if:nF {#1}
1658     {
1659       \centering

```

```

1660         }
1661         \int_gzero:N \g__enumext_minipage_stat_int
1662     }
1663     { \msg_error:nnn { enumext } { wrong-miniright-use } }
1664 % paranoia
1665 \RenewDocumentCommand \miniright { s }
1666 {
1667     \msg_error:nn { enumext } { many-miniright-used }
1668 }
1669 }

```

(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.

```

1670 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1671 {
1672     \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1673     {
1674         \__enumext_keyans_multicols_stop:
1675         \int_compare:nNnT { \l__enumext_columns_v_int } = { 1 }
1676         {
1677             \par\addvspace{ \l__enumext_minipage_after_skip }
1678         }
1679         \end__enumext_mini_page
1680         \hfill
1681         \__enumext_mini_page{ \l__enumext_minipage_right_v_dim }
1682         \par\nointerlineskip
1683         \addvspace { \l__enumext_minipage_right_skip }
1684         \bool_if:nF {#1}
1685         {
1686             \centering
1687         }
1688         \int_gzero:N \g__enumext_minipage_stat_int
1689     }
1690     { \msg_error:nnn { enumext } { wrong-miniright-use } }
1691 % paranoia
1692 \RenewDocumentCommand \miniright { s }
1693 {
1694     \msg_error:nn { enumext } { many-miniright-used }
1695 }
1696 }

```

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

13.24 Setting above and below keys

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

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

```

above* 1697 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
below 1698 {
below* 1699     \keys_define:nn { enumext / #1 }
1700     {
1701         above .skip_set:c = { \l__enumext_vspace_above_#2_skip },
1702         above .value_required:n = true,
1703         above* .code:n = \bool_set_true:c { \l__enumext_vspace_a_star_#2_bool }
1704             \keys_set:nn { enumext / #1 } { above = {##1} },
1705         above* .value_required:n = true,
1706         below .skip_set:c = { \l__enumext_vspace_below_#2_skip },
1707         below .value_required:n = true,
1708         below* .code:n = \bool_set_true:c { \l__enumext_vspace_b_star_#2_bool }
1709             \keys_set:nn { enumext / #1 } { below = {##1} },
1710         below* .value_required:n = true,
1711     }
1712 }
1713 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

(End of definition for above and others.)

13.24.1 Functions for above and below keys in enumext

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

```

1714 \cs_new_protected:Nn \__enumext_vspace_above:
1715 {
1716   \skip_if_eq:nnF
1717   { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1718   {
1719     \bool_if:cTF { \__enumext_vspace_a_star_ \__enumext_level: _bool }
1720     {
1721       \vspace*{ \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1722     }
1723     {
1724       \vspace { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1725     }
1726   }
1727 }

```

(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.

```

1728 \cs_new_protected:Nn \__enumext_vspace_below:
1729 {
1730   \skip_if_eq:nnF
1731   { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1732   {
1733     \bool_if:cTF { \__enumext_vspace_b_star_ \__enumext_level: _bool }
1734     {
1735       \vspace*{ \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1736     }
1737     {
1738       \vspace { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1739     }
1740   }
1741 }

```

(End of definition for `__enumext_vspace_below:`.)

13.24.2 Functions for above and below keys in keyans

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

```

1742 \cs_new_protected:Nn \__enumext_vspace_above_v:
1743 {
1744   \skip_if_eq:nnF { \__enumext_vspace_above_v_skip } { \c_zero_skip }
1745   {
1746     \bool_if:NTF \__enumext_vspace_a_star_v_bool
1747     {
1748       \vspace*{ \__enumext_vspace_above_v_skip }
1749     }
1750     { \vspace { \__enumext_vspace_above_v_skip } }
1751   }
1752 }

```

(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.

```

1753 \cs_new_protected:Nn \__enumext_vspace_below_v:
1754 {
1755   \skip_if_eq:nnF { \__enumext_vspace_below_v_skip } { \c_zero_skip }
1756   {
1757     \bool_if:NTF \__enumext_vspace_b_star_v_bool
1758     {
1759       \vspace*{ \__enumext_vspace_below_v_skip }
1760     }
1761     { \vspace { \__enumext_vspace_below_v_skip } }
1762   }
1763 }

```

(End of definition for `__enumext_vspace_below_v:`.)

13.24.3 Functions for above and below keys in enumext* keyans*

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.

```

1764 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1765 {
1766   \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1767   {
1768     \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1769     {
1770       \vspace*{ \l__enumext_vspace_above_vii_skip }
1771     }
1772     { \vspace { \l__enumext_vspace_above_vii_skip } }
1773   }
1774 }
1775 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1776 {
1777   \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1778   {
1779     \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1780     {
1781       \vspace*{ \l__enumext_vspace_above_viii_skip }
1782     }
1783     { \vspace { \l__enumext_vspace_above_viii_skip } }
1784   }
1785 }

```

(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.

```

1786 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1787 {
1788   \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1789   {
1790     \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1791     {
1792       \vspace*{ \l__enumext_vspace_below_vii_skip }
1793     }
1794     { \vspace { \l__enumext_vspace_below_vii_skip } }
1795   }
1796 }
1797 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1798 {
1799   \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1800   {
1801     \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1802     {
1803       \vspace*{ \l__enumext_vspace_below_viii_skip }
1804     }
1805     { \vspace { \l__enumext_vspace_below_viii_skip } }
1806   }
1807 }

```

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

13.25 Setting series, resume and resume* keys

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

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

```

series
resume
resume*
1808 \cs_set_protected:Npn \__enumext_tmp:n #1
1809 {
1810   \keys_define:nn { enumext / #1 }
1811   {
1812     series .str_set:N = \l__enumext_series_str,
1813     series .value_required:n = true,
1814     resume .code:n = \__enumext_resume_series:n {##1},
1815     resume* .code:n = \__enumext_resume_starred:,

```

```

1816         resume* .value_forbidden:n = true,
1817     }
1818 }
1819 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

13.25.1 Internal functions for series key

The function `__enumext_filter_series:n` will be in charge of filtering the *⟨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
1820 \cs_new:Npn \__enumext_filter_series:n #1
1821 {
1822     \use:e
1823     {
1824         \keyval_parse:NNn
1825         \__enumext_filter_series_key:n
1826         \__enumext_filter_series_pair:nn {#1}
1827     }
1828 }

```

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.

```

1829 \cs_new:Npn \__enumext_filter_series_key:n #1
1830 {
1831     \str_case:nnF {#1}
1832     {
1833         { resume } {} { resume* } {} { base-fix } {}
1834     }
1835     { , { \exp_not:n {#1} } }
1836 }

```

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.

```

1837 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1838 {
1839     \str_case:nnF {#1}
1840     {
1841         { series } {} { resume } {} { start } {}
1842         { start* } {} { save-ans } {} { save-key } {}
1843     }
1844     { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1845 }

```

(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_active_bool` set by the keys `resume` and `resume*` and will call the function `__enumext_resume_last:n`.

🔴 The value of boolean variable `\l__enumext_resume_active_bool` is set to true by the function `__enumext_resume_counter:n` which is used by the keys `resume` and `resume*`, in this case we must Make sure it is set to false so that it does not overwrite the default filtered *⟨keys⟩*. This function is passed to the function `__enumext_parse_keys:n` in the `enumext` environment definition (§13.39) and to the function `__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§13.44).

```

1846 \cs_new_protected:Npn \__enumext_parse_series:n #1
1847 {
1848     \str_if_empty:NTF \l__enumext_series_str
1849     {
1850         \bool_if:NF \l__enumext_resume_active_bool
1851         {
1852             \__enumext_resume_last:n {#1}
1853         }
1854     }
1855     {
1856         \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str _tl }
1857         \tl_gset:ce { g__enumext_series_ \l__enumext_series_str _tl }
1858         { \__enumext_filter_series:n {#1} }
1859         \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str _int }
1860         {

```

```

1861         \int_new:c { g__enumext_series_ \l__enumext_series_str _int }
1862     }
1863 }
1864 }

```

The function `__enumext_resume_last:n` will be in charge of saving the filtering (*keys*) when the *series* key is *not used* and will save them in the variable `\g__enumext_standar_series_tl` for the *enumext* environment and in the variable `\g__enumext_starred_series_tl` for the *enumext** environment.

```

1865 \cs_new_protected:Npn \__enumext_resume_last:n #1
1866 {
1867     \bool_if:NT \l__enumext_standar_first_bool
1868     {
1869         \tl_gclear:N \g__enumext_standar_series_tl
1870         \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1871     }
1872     \bool_if:NT \l__enumext_starred_first_bool
1873     {
1874         \tl_gclear:N \g__enumext_starred_series_tl
1875         \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1876     }
1877 }

```

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

13.25.2 Internal function to save counter value

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

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

```

1878 \cs_new_protected:Npn \__enumext_resume_save_counter:
1879 {
1880     \bool_if:NT \g__enumext_standar_bool
1881     {
1882         \tl_if_empty:NF \l__enumext_series_str
1883         {
1884             \int_gset_eq:cN
1885             { g__enumext_series_ \l__enumext_series_str _int } \value{enumXi}
1886         }
1887         \tl_if_empty:NTF \l__enumext_resume_name_tl
1888         {
1889             \str_if_empty:NT \l__enumext_series_str
1890             {
1891                 \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1892             }
1893         }
1894         {
1895             \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1896             {
1897                 \int_gset_eq:cN
1898                 { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXi}
1899             }
1900         }
1901         \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1902         {
1903             \int_gset_eq:cN
1904             { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1905         }
1906     }
1907     \bool_if:NT \g__enumext_starred_bool
1908     {
1909         \tl_if_empty:NF \l__enumext_series_str
1910         {
1911             \int_gset_eq:cN
1912             { g__enumext_series_ \l__enumext_series_str _int } \value{enumXvii}

```

```

1913     }
1914     \tl_if_empty:NTF \l__enumext_resume_name_tl
1915     {
1916         \str_if_empty:NT \l__enumext_series_str
1917         {
1918             \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1919         }
1920     }
1921     {
1922         \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1923         {
1924             \int_gset_eq:cN
1925             { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXvii}
1926         }
1927     }
1928     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1929     {
1930         \int_gset_eq:cN
1931         { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1932     }
1933 }
1934 }

```

(End of definition for __enumext_resume_save_counter:.)

13.25.3 Internal functions for resume key

__enumext_resume_series:n

The function __enumext_resume_series:n will handle the argument passed to the `resume` key in `enumext` and `enumext*` environments. If the key is passed *without value* the function __enumext_resume_counter: is executed which will set the counter according to the numbering of the last `enumext` or `enumext*` environments in which `series={⟨series name⟩}` key is not present, if the `save-ans` key is active it will set the counter according to the value of the integer variable created by that key, otherwise it will verify that the `\g__enumext_series_⟨series name⟩_tl` variable set by the `series` key exists, if so it will pass these keys to the *first level* of the environment, otherwise it will return an error.

```

1935 \cs_new_protected:Npn \__enumext_resume_series:n #1
1936 {
1937     \tl_if_empty:NTF {#1}
1938     {
1939         \__enumext_resume_counter:n { }
1940     }
1941     {
1942         \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1943         {
1944             \__enumext_resume_counter:n {#1}
1945             \bool_if:NT \g__enumext_standar_bool
1946             {
1947                 \keys_set:nv { enumext / level-1 }
1948                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1949             }
1950             \bool_if:NT \g__enumext_starred_bool
1951             {
1952                 \keys_set:nv { enumext / enumext* }
1953                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1954             }
1955         }
1956         {
1957             \bool_if:NT \g__enumext_standar_bool
1958             {
1959                 \msg_error:nnn { enumext } { unknown-series } {#1}
1960             }
1961             \bool_if:NT \g__enumext_starred_bool
1962             {
1963                 \msg_error:nnn { enumext } { unknown-series } {#1}
1964             }
1965         }
1966     }
1967 }

```

(End of definition for __enumext_resume_series:n)

__enumext_resume_counter:n

__enumext_resume_counter:

__enumext_resume_counter_series:

__enumext_resume_counter_save_anc:

The function __enumext_resume_counter:n will set the variable \l__enumext_resume_active_bool to true and pass the value of the key `resume` to the variable \l__enumext_series_name_tl which will

contain the $\{\langle series\ name\rangle\}$. 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=\{\langle series\ name\rangle\}` we will execute the function `__enumext_resume_counter_series:`, finally we will execute the function `__enumext_resume_counter_save_ans:` which is associated with the key *save-ans*.

```

1968 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1969 {
1970   \bool_set_true:N \l__enumext_resume_active_bool
1971   \tl_set:Nn \l__enumext_resume_name_tl {#1}
1972   \tl_if_empty:NTF \l__enumext_resume_name_tl
1973   {
1974     \__enumext_resume_counter:
1975   }
1976   {
1977     \__enumext_resume_counter_series:
1978   }
1979   \__enumext_resume_counter_save_ans:
1980 }

```

The `__enumext_resume_counter:` function is executed when the *resume* key is used *without value*, only the counters for the “*first level*” of the environments will be set.

```

1981 \cs_new_protected:Nn \__enumext_resume_counter:
1982 {
1983   \bool_if:NT \g__enumext_standar_bool
1984   {
1985     \int_gincr:N \g__enumext_resume_int
1986     \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1987   }
1988   \bool_if:NT \g__enumext_starred_bool
1989   {
1990     \int_gincr:N \g__enumext_resume_vii_int
1991     \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1992   }
1993 }

```

The function `__enumext_resume_counter_series:` will be executed when the `resume=\{\langle series\ name\rangle\}` key is active, setting the counters for the “*first level*” of the environments according to the value of the integer variables created by the *series* key.

```

1994 \cs_new_protected:Nn \__enumext_resume_counter_series:
1995 {
1996   \bool_if:NT \g__enumext_standar_bool
1997   {
1998     \int_set:Nn \l__enumext_start_i_int
1999     {
2000       \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
2001     }
2002   }
2003   \bool_if:NT \g__enumext_starred_bool
2004   {
2005     \int_set:Nn \l__enumext_start_vii_int
2006     {
2007       \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
2008     }
2009   }
2010 }

```

The function `__enumext_resume_counter_save_ans:` will be executed when the *save-ans* key is active along with the *resume* key, setting the counters for the “*first level*” of the environments according to the value of the integer variables created by the *save-ans* key.

```

2011 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
2012 {
2013   \bool_lazy_and:nnT
2014   { \bool_if_p:N \l__enumext_standar_first_bool }
2015   { \bool_if_p:N \l__enumext_store_active_bool }
2016   {
2017     \int_set:Nn \l__enumext_start_i_int
2018     {
2019       \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
2020     }
2021   }
2022   \bool_lazy_and:nnT

```

```

2023     { \bool_if_p:N \__enumext_starred_first_bool }
2024     { \bool_if_p:N \__enumext_store_active_bool }
2025     {
2026         \int_set:Nn \__enumext_start_vii_int
2027         {
2028             \int_use:c { g__enumext_resume_ \__enumext_store_name_tl _int } + 1
2029         }
2030     }
2031 }

```

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

13.25.4 Internal function for resume* key

`__enumext_resume_starred:` The function `__enumext_resume_starred:` will handle the `resume*` key in the `enumext` and `enumext*` environments. This function will execute the filtered `(keys)` in the last one and will continue with the numbering according to the last execution of the environment `enumext` or `enumext*` in which the keys `resume={⟨series name⟩}` or `series={⟨series name⟩}` were not active.

```

2032 \cs_new_protected:Nn \__enumext_resume_starred:
2033 {
2034     \bool_if:NT \g__enumext_standar_bool
2035     {
2036         \tl_if_empty:NF \g__enumext_standar_series_tl
2037         {
2038             \__enumext_resume_counter:n { }
2039             \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
2040         }
2041     }
2042     \bool_if:NT \g__enumext_starred_bool
2043     {
2044         \tl_if_empty:NF \g__enumext_starred_series_tl
2045         {
2046             \__enumext_resume_counter:n { }
2047             \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
2048         }
2049     }
2050 }

```

(End of definition for `__enumext_resume_starred:`.)

13.26 Setting save-ans, check-ans and no-store keys

The key `save-ans` is directly associated with the keys `check-ans`, `no-store`, `resume` and `resume*`, this will activate the entire “storage system” in the `enumext` package.

13.26.1 Setting save-ans key

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

```

2051 \cs_set_protected:Npn \__enumext_tmp:n #1
2052 {
2053     \keys_define:nn { enumext / #1 }
2054     {
2055         save-ans .code:n = \__enumext_storing_set:n {##1},
2056         save-ans .value_required:n = true,
2057     }
2058 }
2059 \clist_map_inline:nn { level-1, enumext* } { { \__enumext_tmp:n {#1} } }

```

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

13.26.2 Internal functions for save-ans key

`__enumext_start_save_ans_msg:` 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:`.

```

2060 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
2061 {
2062     \msg_term:nnVV { enumext } { save-ans-log }
2063     \g__enumext_envir_name_tl \__enumext_store_name_tl
2064 }
2065 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
2066 {

```



```

2067     \msg_term:nnVV { enumext } { save-ans-log-hook }
2068     \g__enumext_envir_name_tl \g__enumext_store_name_tl
2069 }

```

(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.

```

2070 \cs_new_protected:Npn \__enumext_storing_set:n #1
2071 {
2072     \tl_set:Nx \l__enumext_store_name_tl {#1}
2073     \tl_if_empty:NTF \l__enumext_store_name_tl
2074     {
2075         \bool_lazy_or:nnT
2076         { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
2077         {
2078             \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
2079         }
2080     }
2081     {
2082         \bool_lazy_or:nnT
2083         { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
2084         {
2085             \__enumext_start_save_ans_msg:
2086             \__enumext_storing_exec:
2087         }
2088     }
2089 }

```

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.

```

2090 \cs_new_protected:Nn \__enumext_storing_exec:
2091 {
2092     \bool_set_true:N \l__enumext_store_active_bool
2093     \bool_set_true:N \l__enumext_check_answers_bool
2094     \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*`.

```

2095     \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
2096     {
2097         \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
2098         \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
2099     }
2100     \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
2101     {
2102         \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
2103         \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
2104     }
2105     \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
2106     {
2107         \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
2108         \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
2109     }
2110 }

```

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

13.26.3 The check answer mechanism

The internal mechanism for “checking answers” follows this logic:

If the line begins with `\item` or `\item*` and does NOT *open a nested environment*, each `\item` or `\item*` must contain a *single* execution of the `\anskey` command, i.e. the counter of the

executions of the `\anskey` command must be equal to the counter associated with the sum of executions of `\item` and `\item*`.

If the line begins with `\item` or `\item*` and opens a nested environment each `\item` or `\item*` in the nested environment must have a *single* execution of the `\anskey` command and the counter associated to the sum of `\item` and `\item*` executions must decrementing by “one” to maintain equality.

In order for the mechanism for the check-answer to work (not counting `keyans`, `keyans*` and `keyanspic`) we need:

1. We must keep track of the total number of `\item` and `\item*` (enumerated) that appear within the environment including the nested levels.
2. We must keep track of the total number of `\item` and `\item*` (enumerated) that appear per level of nesting.
3. Keeping track of the number of times the environment nests.

The integer variable associated to the sum of each `\item` and `\item*` in the environment `\g__enumext_item_number_int` must match the integer variable `\g__enumext_item_anskey_int` associated to the execution of the command `\anskey`. We analyze the cases:

- a) If the list only has one level the number of `\item` + `\item*` = `\anskey`
- b) If the list has *nested levels*, for each level of nesting we need to decrementing by one (for the `\item` or `\item*` that opens the nest) so that the account remains the same.

With `keyans`, `keyans*` and `keyanspic` it is enough to increase in one the integer of `\anskey`. The integers created must be global if they are not lost in the interior levels of nesting and to execute the test we will use a “hook” function after closing the *first level* of the environment.

13.26.4 Setting check-ans and no-store keys

Now we define the keys `check-ans` and `no-store` for all levels of `enumext` and `enumext*` environments.

```
check-ans 2111 \cs_set_protected:Npn \__enumext_tmp:n #1
no-store 2112 {
2113   \keys_define:nn { enumext / #1 }
2114   {
2115     check-ans .bool_set:N = \__enumext_check_ans_key_bool,
2116     check-ans .initial:n = false,
2117     check-ans .value_required:n = true,
2118     no-store .code:n = {
2119       \bool_set_false:N \__enumext_check_answers_bool
2120       \bool_set_false:N \__enumext_check_ans_key_bool
2121     },
2122     no-store .value_forbidden:n = true,
2123   }
2124 }
2125 \clist_map_inline:nn
2126 {
2127   level-1, level-2, level-3, level-4, enumext*
2128 }
2129 { \__enumext_tmp:n {#1} }
```

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

13.26.5 Set-up check answer mechanism

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

```
2130 \cs_new_protected:Nn \__enumext_check_ans_active:
2131 {
2132   \tl_if_empty:NF \l__enumext_store_name_tl
2133   {
2134     \bool_if:NT \l__enumext_check_answers_bool
2135     {
2136       \__enumext_check_ans_level:
2137     }
2138   }
2139 }
```

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”.

```

2140 \cs_new_protected:Nn \__enumext_check_ans_level:
2141 {
2142   \int_case:nn { \__enumext_level_int }
2143   {
2144     { 1 }{
2145       \bool_lazy_all:nT
2146       {
2147         { \bool_if_p:N \g__enumext_starred_bool }
2148         { \int_compare_p:nNn { \__enumext_level_h_int } = { 1 } }
2149       }
2150       {
2151         \int_gdecr:N \g__enumext_item_number_int
2152         \bool_set_false:N \l__enumext_item_number_bool
2153       }
2154     }
2155     { 2 }{
2156       \int_gdecr:N \g__enumext_item_number_int
2157       \bool_set_false:N \l__enumext_item_number_bool
2158     }
2159     { 3 }{
2160       \int_gdecr:N \g__enumext_item_number_int
2161       \bool_set_false:N \l__enumext_item_number_bool
2162     }
2163     { 4 }{
2164       \int_gdecr:N \g__enumext_item_number_int
2165       \bool_set_false:N \l__enumext_item_number_bool
2166     }
2167   }

```

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.

```

2168   \int_case:nn { \__enumext_level_h_int }
2169   {
2170     { 1 }{
2171       \bool_lazy_all:nT
2172       {
2173         { \bool_if_p:N \g__enumext_standar_bool }
2174         { \int_compare_p:nNn { \__enumext_level_int } = { 1 } }
2175       }
2176       {
2177         \int_gdecr:N \g__enumext_item_number_int
2178         \bool_set_false:N \l__enumext_item_number_bool
2179       }
2180     }
2181   }
2182 }

```

(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.

```

2183 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
2184 {
2185   \bool_lazy_and:nnT
2186   { \bool_if_p:N \l__enumext_check_ans_key_bool }
2187   { \bool_if_p:N \g__enumext_standar_bool }
2188   {
2189     \bool_gset_true:N \g__enumext_check_ans_key_bool
2190   }
2191   \bool_lazy_and:nnT
2192   { \bool_if_p:N \l__enumext_check_ans_key_bool }
2193   { \bool_if_p:N \g__enumext_starred_bool }
2194   {
2195     \bool_gset_true:N \g__enumext_check_ans_key_bool
2196   }
2197 }

```

(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:`.

```

2198 \cs_new_protected:Nn \__enumext_item_answer_diff:
2199 {
2200   \int_gset:Nn \g__enumext_item_answer_diff_int
2201   {
2202     \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
2203   }
2204 }

```

(End of definition for `__enumext_item_answer_diff:`.)

```

\__enumext_check_ans_show:
  \__enumext_check_ans_msg_less:
  \__enumext_check_ans_msg_same_ok:
  \__enumext_check_ans_msg_greater:

```

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:`.

```

2205 \cs_new_protected:Nn \__enumext_check_ans_show:
2206 {
2207   \int_case:nn { \g__enumext_item_answer_diff_int }
2208   {
2209     { -1 } { \__enumext_check_ans_msg_less:    }
2210     {  0 } { \__enumext_check_ans_msg_same_ok:  }
2211     {  1 } { \__enumext_check_ans_msg_greater:  }
2212   }
2213 }
2214 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
2215 {
2216   \msg_warning:nnee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2217   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2218 }
2219 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
2220 {
2221   \msg_term:nnee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2222   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2223 }
2224 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
2225 {
2226   \msg_warning:nnee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2227   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2228 }

```

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

```

\__enumext_check_ans_log:
  \__enumext_check_ans_log_msg_less:
  \__enumext_check_ans_log_msg_same_ok:
  \__enumext_check_ans_log_msg_greater:

```

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:`.

```

2229 \cs_new_protected:Nn \__enumext_check_ans_log:
2230 {
2231   \int_case:nn { \g__enumext_item_answer_diff_int }
2232   {
2233     { -1 } { \__enumext_check_ans_log_msg_less:    }
2234     {  0 } { \__enumext_check_ans_log_msg_same_ok:  }
2235     {  1 } { \__enumext_check_ans_log_msg_greater:  }
2236   }
2237 }
2238 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
2239 {
2240   \msg_log:nnee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2241   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2242 }
2243 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
2244 {
2245   \msg_log:nnee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2246   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2247 }
2248 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
2249 {
2250   \msg_log:nnee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2251   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2252 }

```

(End of definition for `__enumext_check_ans_log`; and others.)

13.26.6 Check for `\item*` and `\anspic*` commands

`__enumext_check_starred_cmd:n`

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

```

2253 \cs_new_protected:Npn __enumext_check_starred_cmd:n #1
2254 {
2255   \int_compare:nNtT
2256     { \g__enumext_check_starred_cmd_int } = { 0 }
2257   {
2258     \msg_warning:nnnV
2259       { enumext } { missing-starred }{ #1 } \l__enumext_check_start_line_env_tl
2260   }
2261   \int_compare:nNtT
2262     { \g__enumext_check_starred_cmd_int } > { 1 }
2263   {
2264     \msg_warning:nnnV
2265       { enumext } { many-starred }{ #1 } \l__enumext_check_start_line_env_tl
2266   }
2267   \int_gzero:N \g__enumext_check_starred_cmd_int
2268   \tl_clear:N \l__enumext_check_start_line_env_tl
2269 }

```

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

13.27 Keys and functions associated with storage

13.27.1 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`.

```

2270 \cs_set_protected:Npn __enumext_tmp:nn #1 #2
2271 {
2272   \keys_define:nn { enumext / #1 }
2273   {
2274     mark-ans* .tl_set:c = { \l__enumext_mark_answer_sym_#2_tl },
2275     mark-ans* .initial:n = \textasteriskcentered,
2276     mark-ans* .value_required:n = true,
2277     mark-pos* .choice:,
2278     mark-pos* / left .code:n = \str_set:cn { \l__enumext_mark_position_#2_str } { l },
2279     mark-pos* / right .code:n = \str_set:cn { \l__enumext_mark_position_#2_str } { r },
2280     mark-pos* / center .code:n = \str_set:cn { \l__enumext_mark_position_#2_str } { c },
2281     mark-pos* / unknown .code:n =
2282       \msg_error:nnee { enumext } { unknown-choice }
2283       { mark-pos } { left,~right,~center } { \exp_not:n {##1} },
2284     mark-pos* .initial:n = right,
2285     mark-pos* .value_required:n = true,
2286     mark-sep* .dim_set:c = { \l__enumext_mark_sym_sep_#2_dim },
2287     mark-sep* .value_required:n = true,
2288     wrap-ans* .cs_set_protected:cp = { __enumext_keyans_wrapper_item_#2:n } ##1,
2289     wrap-ans* .value_required:n = true,
2290     wrap-opt .cs_set_protected:cp = { __enumext_keyans_wrapper_opt_#2:n } ##1,
2291     wrap-opt .initial:n = [{##1}],
2292     wrap-opt .value_required:n = true,
2293     save-sep .tl_set:c = { \l__enumext_store_keyans_item_opt_sep_#2_tl },
2294     save-sep .initial:n = {,~},
2295     save-sep .value_required:n = true,
2296     show-ans .bool_set:N = \l__enumext_show_answer_bool,
2297     show-ans .initial:n = false,
2298     show-ans .value_required:n = true,
2299     show-pos .bool_set:N = \l__enumext_show_position_bool,
2300     show-pos .initial:n = false,
2301     show-pos .value_required:n = true,
2302   }
2303 }
2304 \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*`.

```

2305 \cs_set_protected:Npn \__enumext_tmp:n #1
2306 {
2307   \keys_define:nn { enumext / #1 }
2308   {
2309     mark-ref .tl_set:N = \__enumext_mark_ref_sym_tl,
2310     mark-ref .initial:n = \textreferencemark,
2311     mark-ref .value_required:n = true,
2312     save-ref .bool_set:N = \__enumext_store_ref_key_bool,
2313     save-ref .initial:n = false,
2314     save-ref .value_required:n = true,
2315     show-ans .bool_set:N = \__enumext_show_answer_bool,
2316     show-ans .initial:n = false,
2317     show-ans .value_required:n = true,
2318     show-pos .bool_set:N = \__enumext_show_position_bool,
2319     show-pos .initial:n = false,
2320     show-pos .value_required:n = true,
2321     mark-ans .tl_set:N = \__enumext_mark_answer_sym_tl,
2322     mark-ans .initial:n = \textasteriskcentered,
2323     mark-ans .value_required:n = true,
2324     mark-sep .dim_set:N = \__enumext_mark_sym_sep_dim,
2325     mark-sep .value_required:n = true,
2326     mark-pos .choice:,
2327     mark-pos / left .code:n = \str_set:Nn \__enumext_mark_position_str { l },
2328     mark-pos / right .code:n = \str_set:Nn \__enumext_mark_position_str { r },
2329     mark-pos / center .code:n = \str_set:Nn \__enumext_mark_position_str { c },
2330     mark-pos / unknown .code:n =
2331       \msg_error:nneee { enumext } { unknown-choice }
2332       { mark-pos } { left,~right,~center } { \exp_not:n {##1} },
2333     mark-pos .initial:n = right,
2334     mark-pos .value_required:n = true,
2335
2336     wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
2337     wrap-ans .initial:n =
2338       {
2339         \fbox{\parbox[t]{\dimeval{\itemwidth -2\fboxsep -2\fboxrule}}{##1}}
2340       },
2341     wrap-ans .value_required:n = true,
2342     mark-ans* .code:n = {
2343       \keys_set:nn { enumext / keyans } { mark-ans* = {##1} }
2344       \keys_set:nn { enumext / keyans* } { mark-ans* = {##1} }
2345     },
2346     mark-ans* .value_required:n = true,
2347     mark-pos* .code:n = {
2348       \keys_set:nn { enumext / keyans } { mark-pos* = {##1} }
2349       \keys_set:nn { enumext / keyans* } { mark-pos* = {##1} }
2350     },
2351     mark-pos* .value_required:n = true,
2352     mark-sep* .code:n = {
2353       \keys_set:nn { enumext / keyans } { mark-sep* = {##1} }
2354       \keys_set:nn { enumext / keyans* } { mark-sep* = {##1} }
2355     },
2356     mark-sep* .value_required:n = true,
2357     wrap-ans* .code:n = {
2358       \keys_set:nn { enumext / keyans } { wrap-ans* = {##1} }
2359       \keys_set:nn { enumext / keyans* } { wrap-ans* = {##1} }
2360     },
2361     wrap-ans* .value_required:n = true,
2362     wrap-opt .code:n = {
2363       \keys_set:nn { enumext / keyans } { wrap-opt = {##1} }
2364       \keys_set:nn { enumext / keyans* } { wrap-opt = {##1} }
2365     },
2366     wrap-opt .value_required:n = true,
2367     save-sep .code:n = {
2368       \keys_set:nn { enumext / keyans } { save-sep = {##1} }
2369       \keys_set:nn { enumext / keyans* } { save-sep = {##1} }
2370     },
2371     save-sep .value_required:n = true,

```



```

2372     }
2373   }
2374   \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

13.27.2 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*.

```

2375 \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2376 {
2377   \bool_if:cF { \__enumext_store_save_key_ \__enumext_level: _bool }
2378   {
2379     \tl_clear:c { \__enumext_store_save_key_ \__enumext_level: _tl }
2380     \tl_set:ce
2381       { \__enumext_store_save_key_ \__enumext_level: _tl }
2382       { \__enumext_filter_save_key:n {#1} }
2383   }
2384 }
2385 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2386 {
2387   \bool_if:NF \__enumext_store_save_key_vii_bool
2388   {
2389     \tl_clear:N \__enumext_store_save_key_vii_tl
2390     \tl_set:Ne \__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2391   }
2392 }

```

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

13.27.3 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.

```

2393 \cs_set_protected:Npn \__enumext_tmp:n #1
2394 {
2395   \keys_define:nn { enumext / enumext* }
2396   {
2397     save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2398     save-key .value_required:n = true,
2399   }
2400   \keys_define:nn { enumext / #1 }
2401   {
2402     save-key .code:n = \__enumext_parse_save_key:n {##1},
2403     save-key .value_required:n = true,
2404   }
2405 }
2406 \clist_map_inline:nn { level-1, level-2, level-3, level-4 } { \__enumext_tmp:n {#1} }

```

(End of definition for *save-key*.)

```

\__enumext_parse_save_key:n
\__enumext_parse_save_key_vii:n

```

The functions `__enumext_parse_save_key:n` and `__enumext_parse_save_key_vii:n` will be responsible for “*storing keys*” in the variable `__enumext_store_save_key_X_tl` for `enumext` and `enumext*`.

```

2407 \cs_new_protected:Npn \__enumext_parse_save_key:n #1
2408 {
2409   \bool_set_true:c { \__enumext_store_save_key_ \__enumext_level: _bool }
2410   \tl_clear:c { \__enumext_store_save_key_ \__enumext_level: _tl }

```

```

2411 \tl_set:ce
2412 { \__enumext_store_save_key_ \__enumext_level: _tl }
2413 { \__enumext_filter_save_key:n {#1} }
2414 }
2415 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2416 {
2417   \bool_set_true:N \__enumext_store_save_key_vii_bool
2418   \tl_clear:N \__enumext_store_save_key_vii_tl
2419   \tl_set:Nx \__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2420 }

```

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

13.27.4 Internal functions to store optional arguments

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

```

2421 \cs_new:Npn \__enumext_filter_save_key:n #1
2422 {
2423   \use:e
2424   {
2425     \keyval_parse:NNn
2426     \__enumext_filter_save_key_key:n
2427     \__enumext_filter_save_key_pair:nn {#1}
2428   }
2429 }

```

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.

```

2430 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2431 {
2432   \str_case:nnF {#1}
2433   {
2434     { resume } {} { resume* } {} { no-store } {} { base-fix } {}
2435   }
2436   { , { \exp_not:n {#1} } }
2437 }

```

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.

```

2438 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2439 {
2440   \str_case:nnF {#1}
2441   {
2442     { series } {} { resume } {} { save-ans } {} { save-ref } {}
2443     { save-key } {} { check-ans } {} { show-ans } {} { show-pos } {}
2444     { mark-ans } {} { mark-pos } {} { mark-sep } {} { wrap-ans } {}
2445     { mark-ans* } {} { mark-pos* } {} { mark-sep* } {} { wrap-ans* } {}
2446     { wrap-opt } {} { save-sep } {} { mark-ref } {} { mini-env } {}
2447     { mini-sep } {} { mini-right } {} { mini-right* } {}
2448   }
2449   { , { \exp_not:n {#1} } } = { \exp_not:n {#2} } }
2450 }

```

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

13.27.5 Function for storing content in prop list

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

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

```

2451 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2452 {
2453   \prop_gput_if_not_in:cen { g__enumext_ \__enumext_store_name_tl _prop }
2454   {
2455     \int_eval:n { \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop } + 1 }
2456   }
2457   { #1 }

```

```

2458     }
2459     \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V }

```

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

13.27.6 Function for storing content in sequence

```

\__enumext_store_addto_seq:n
\__enumext_store_addto_seq:v
\__enumext_store_addto_seq:V

```

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.

```

2460 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2461 {
2462     \seq_gput_right:cn { g__enumext_ \__enumext_store_name_tl_seq } { #1 }
2463 }
2464 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V }

```

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

13.27.7 Functions for storing structure in the sequence

```

\__enumext_store_level_open:
\__enumext_store_level_close:

```

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.

```

2465 \cs_new_protected:Nn \__enumext_store_level_open:
2466 {
2467     \bool_if:NT \__enumext_check_answers_bool
2468     {
2469         \tl_if_empty:cTF { \__enumext_store_save_key_ \__enumext_level: _tl }
2470         {
2471             \__enumext_store_addto_seq:n
2472             {
2473                 \item \begin{enumext}
2474             }
2475         }
2476         {
2477             \tl_put_left:cn { \__enumext_store_save_key_ \__enumext_level: _tl }
2478             {
2479                 \item \begin{enumext} [
2480             }
2481             \tl_put_right:cn { \__enumext_store_save_key_ \__enumext_level: _tl }
2482             {
2483                 ]
2484             }
2485             \__enumext_store_addto_seq:v { \__enumext_store_save_key_ \__enumext_level: _tl }
2486         }
2487     }
2488 }
2489 \cs_new_protected:Nn \__enumext_store_level_close:
2490 {
2491     \bool_if:NT \__enumext_check_answers_bool
2492     {
2493         \__enumext_store_addto_seq:n { \end{enumext} }
2494     }
2495 }

```

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

```

\__enumext_store_level_open_vii:
\__enumext_store_level_close_vii:

```

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.

```

2496 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2497 {
2498     \bool_if:NT \__enumext_check_answers_bool
2499     {
2500         \tl_if_empty:NTF \__enumext_store_save_key_vii_tl
2501         {
2502             \__enumext_store_addto_seq:n
2503             {
2504                 \item \begin{enumext*}
2505             }
2506         }
2507         {

```

```

2508         \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2509         {
2510             \item \begin{enumext*}[
2511             ]
2512         \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2513         {
2514             ]
2515         }
2516         \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2517     }
2518 }
2519 }
2520 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2521 {
2522     \bool_if:NT \l__enumext_check_answers_bool
2523     {
2524         \__enumext_store_addto_seq:n { \end{enumext*} }
2525     }
2526 }

```

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

13.27.8 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`

```

2527 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2528 {
2529     \mode_leave_vertical:
2530     \skip_horizontal:n { -\dim_use:N #2 }
2531     \hbox_overlap_left:n
2532     {
2533         \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2534         {
2535             \tl_use:N \l__enumext_mark_answer_sym_tl
2536         }
2537     }
2538     \skip_horizontal:n { \dim_use:N #2 }
2539 }
2540 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }

```

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

13.28 The internal label and ref

The function `__enumext_store_internal_ref:` handles the “*internal label and ref*” system used by the `save-ref` and `mark-ref` keys for `\anskey` will allow to execute `\ref{⟨store name : position⟩}` and will return `1.(a).i.A`.

```

\__enumext_store_internal_ref:

```

First we will remove the dots “.” from the current `⟨labels⟩`, we do not want to get double dots in our references, then we will place this in the variable `\l__enumext_newlabel_arg_two_tl`.

```

2541 \cs_new_protected:Nn \__enumext_store_internal_ref:
2542 {
2543     \cs_set_protected:Npn \__enumext_tmp:n ##1
2544     {
2545         \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2546         \tl_reverse:c { l__enumext_label_copy_##1_tl }
2547         \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2548         \tl_reverse:c { l__enumext_label_copy_##1_tl }
2549     }
2550     \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2551     \cs_set:Npn \__enumext_tmp:n ##1
2552     { . \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.

```

2553     \bool_lazy_all:nT
2554     {
2555         { \bool_if_p:N \g__enumext_starred_bool }
2556         { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }

```

```

2557     }
2558     {
2559         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2560         { \tl_use:N \l__enumext_label_copy_vii_tl }
2561     }
2562     \bool_lazy_all:nT
2563     {
2564         { \bool_not_p:n { \g__enumext_standar_bool } }
2565         { \bool_if_p:N \l__enumext_standar_bool }
2566         { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2567     }
2568     {
2569         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2570         {
2571             \tl_use:N \l__enumext_label_copy_vii_tl
2572             \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2573         }
2574     }

```

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.

```

2575     \bool_lazy_all:nT
2576     {
2577         { \bool_if_p:N \g__enumext_standar_bool }
2578         { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2579         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2580     }
2581     {
2582         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2583         {
2584             \tl_use:N \l__enumext_label_copy_i_tl
2585             \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2586         }
2587     }
2588     \cs_set:Npn \__enumext_tmp:n ##1
2589     { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } . }
2590     \bool_lazy_all:nT
2591     {
2592         { \bool_if_p:N \g__enumext_standar_bool }
2593         { \bool_if_p:N \l__enumext_starred_bool }
2594         { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2595     }
2596     {
2597         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2598         {
2599             \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2600             \tl_use:N \l__enumext_label_copy_vii_tl
2601         }
2602     }

```

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

```

2603     \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2604     {
2605         \l__enumext_store_name_tl \c_colon_str
2606         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2607     }

```

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

```

2608     \tl_put_right:Ne \l__enumext_write_aux_file_tl
2609     {
2610         \__enumext_newlabel:nn
2611         { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2612         { \l__enumext_newlabel_arg_two_tl }
2613     }
2614     \l__enumext_write_aux_file_tl
2615 }

```

(End of definition for `__enumext_store_internal_ref:`)

13.29 Common functions for \anskey and anskey* environment

__enumext_store_anskey_arg:n

The internal function __enumext_store_anskey_arg:n first we pass the $\{\langle argument \rangle\}$ 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*” $\{\langle argument \rangle\}$.

```

2616 \cs_new_protected:Npn \__enumext_store_anskey_arg:n #1
2617 {
2618   \int_gincr:N \g__enumext_item_anskey_int
2619   \__enumext_store_addto_prop:n {#1}
2620   \bool_if:NT \l__enumext_store_ref_key_bool
2621   {
2622     \__enumext_store_internal_ref:
2623   }
2624   \__enumext_anskey_show_wrap_left:n { #1 }

```

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

```

2625   \tl_clear:N \l__enumext_store_anskey_arg_tl
2626   \bool_lazy_and:nnT
2627   { \bool_if_p:N \l__enumext_store_columns_break_bool }
2628   { \bool_not_p:n { \l__enumext_starred_bool } }
2629   {
2630     \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2631   }
2632   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { \item }

```

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

```

2633   \bool_lazy_and:nnT
2634   { \bool_not_p:n { \l__enumext_starred_bool } }
2635   { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2636   {
2637     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2638     {
2639       ( \exp_not:V \l__enumext_store_item_join_int )
2640     }
2641   }

```

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

```

2642   \bool_if:NTF \l__enumext_store_item_star_bool
2643   {
2644     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2645     \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2646     {
2647       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2648       {
2649         [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2650       }
2651     }
2652     \dim_compare:nT
2653     {
2654       \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2655     }
2656     {
2657       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2658       {
2659         [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2660       }
2661     }
2662     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2663   }
2664   {
2665     \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2666   }

```

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*.

```

2667   \bool_lazy_and:nnT

```

```

2668 { \bool_if_p:N \l__enumext_store_ref_key_bool }
2669 { \bool_if_p:N \l__enumext_hyperref_bool }
2670 {
2671   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2672   {
2673     \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2674     { \exp_not:V \l__enumext_mark_ref_sym_tl }
2675   }
2676 }
2677 \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2678 }

```

(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.

```

2679 \cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
2680 {
2681   \par
2682   \bool_if:NTF \l__enumext_starred_bool
2683   {
2684     \dim_compare:nNtT { \l__enumext_mark_sym_sep_dim } = { \c_zero_dim }
2685     {
2686       \dim_set:Nn \l__enumext_mark_sym_sep_dim { \l__enumext_labelsep_vii_dim }
2687     }
2688     \__enumext_print_keyans_box:NN
2689     \l__enumext_labelwidth_vii_dim \l__enumext_mark_sym_sep_dim
2690   }
2691   {
2692     \dim_compare:nNtT { \l__enumext_mark_sym_sep_dim } = { \c_zero_dim }
2693     {
2694       \dim_set:Nn \l__enumext_mark_sym_sep_dim
2695       {
2696         \dim_use:c { \l__enumext_labelsep_ \__enumext_level: _dim }
2697       }
2698     }
2699     \__enumext_print_keyans_box:cc
2700     { \l__enumext_labelwidth_ \__enumext_level: _dim } { \l__enumext_mark_sym_sep_dim }
2701   }
2702   \__enumext_anskey_wrapper:n { #1 }
2703 }

```

(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.

```

2704 \cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
2705 {
2706   \bool_if:NT \l__enumext_show_answer_bool
2707   {
2708     \__enumext_anskey_show_wrap_arg:n { #1 }
2709   }
2710   \bool_if:NT \l__enumext_show_position_bool
2711   {
2712     \tl_set:Ne \l__enumext_mark_answer_sym_tl
2713     {
2714       \group_begin:
2715       \exp_not:N \normalfont
2716       \exp_not:N \footnotesize [ \int_eval:n
2717       {
2718         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2719       }
2720       ]
2721       \group_end:
2722     }
2723     \__enumext_anskey_show_wrap_arg:n { #1 }
2724   }
2725 }

```

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

13.30 The command \anskey

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

The `\anskey` command will cover this point and give it similar behaviour to that of `\item` in the `enumext` and `enumext*` environments executed as follows `\anskey[⟨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:n

2726 \keys_define:nn { enumext / anskey }
2727 {
2728   break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2729   break-col .default:n = true,
2730   break-col .value_forbidden:n = true,
2731   item-join .int_set:N = \l__enumext_store_item_join_int,
2732   item-join .value_required:n = true,
2733   item-star .bool_set:N = \l__enumext_store_item_star_bool,
2734   item-star .default:n = true,
2735   item-star .value_forbidden:n = true,
2736   item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
2737   item-sym* .value_required:n = true,
2738   item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
2739   item-pos* .value_required:n = true,
2740   unknown .code:n = { \__enumext_anskey_unknown:n {#1} },
2741 }
```

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`.

```
2742 \cs_new_protected:Npn \__enumext_anskey_unknown:n #1
2743 {
2744   \exp_args:NV \__enumext_anskey_unknown:nn \l_keys_key_str {#1}
2745 }
2746 \cs_new_protected:Npn \__enumext_anskey_unknown:nn #1 #2
2747 {
2748   \tl_if_blank:nTF {#2}
2749   {
2750     \msg_error:nnn { enumext } { anskey-cmd-key-unknown } {#1}
2751   }
2752   {
2753     \msg_error:nnnn { enumext } { anskey-cmd-key-value-unknown } {#1} {#2}
2754   }
2755 }
```

(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`.

```
2756 \NewDocumentCommand \anskey { o +m }
2757 {
2758   \__enumext_anskey_safe_outer:
2759   \group_begin:
2760     \bool_if:NT \l__enumext_check_answers_bool
2761     {
2762       \tl_if_novalue:nF {#1}
2763       {
2764         \keys_set:nn { enumext / anskey } {#1}
2765       }
2766       \tl_if_blank:nTF {#2}
2767       {
2768         \msg_error:nn { enumext } { anskey-empty-arg }
2769       }
2770       {
2771         \__enumext_anskey_safe_inner:
2772         \__enumext_store_anskey_arg:n {#2}
2773       }
2774     }
```

```

2775   \group_end:
2776 }

```

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

13.30.1 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.

```

2777 \cs_new_protected:Nn \__enumext_anskey_safe_outer:
2778 {
2779   \bool_if:NF \l__enumext_store_active_bool
2780   {
2781     \msg_error:nnnn { enumext } { anskey-wrong-place } { anskey } { enumext }
2782   }
2783   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2784   {
2785     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans }
2786   }
2787   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2788   {
2789     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans* }
2790   }
2791   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2792   {
2793     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyanspic }
2794   }
2795 }

```

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.

```

2796 \cs_new_protected:Nn \__enumext_anskey_safe_inner:
2797 {
2798   \int_incr:N \l__enumext_anskey_level_int
2799   \int_compare:nNnT { \l__enumext_anskey_level_int } > { 1 }
2800   {
2801     \msg_error:nn { enumext } { anskey-nested }
2802   }
2803   \bool_if:NF \l__enumext_item_number_bool
2804   {
2805     \msg_error:nn { enumext } { anskey-unnumber-item }
2806   }
2807   \mode_if_math:T
2808   {
2809     \msg_error:nne { enumext } { anskey-math-mode } { \c_backslash_str anskey }
2810   }
2811 }

```

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

13.31 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 `ltxcmd`[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.

```

2812 \keys_define:nn { enumext / anskey* }
2813 {
2814   break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2815   break-col .default:n = true,
2816   break-col .value_forbidden:n = true,
2817   item-join .int_set:N = \l__enumext_store_item_join_int,
2818   item-join .value_required:n = true,
2819   item-star .bool_set:N = \l__enumext_store_item_star_bool,
2820   item-star .default:n = true,
2821   item-star .value_forbidden:n = true,
2822   item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
2823   item-sym* .value_required:n = true,
2824   item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,

```

```

2825     item-pos* .value_required:n = true,
2826     force-eol .bool_set:N = \l__enumext_anskey_env_force_eol_bool,
2827     force-eol .initial:n = false,
2828     force-eol .default:n = true,
2829     write-env .code:n = {
2830         \bool_set_true:N \l__enumext_write_anskey_env_bool
2831         \tl_set:Nn \l__enumext_write_anskey_env_file_name_tl {#1}
2832     },
2833     write-env .value_required:n = true,
2834     overwrite .bool_set:N = \l__enumext_anskey_env_overwrite_bool,
2835     overwrite .initial:n = false,
2836     overwrite .default:n = true,
2837     unknown .code:n = { \l__enumext_anskey_env_unknown:n {#1} },
2838 }

```

(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`.

```

2839 \cs_new_protected:Npn \__enumext_anskey_env_unknown:n #1
2840 {
2841     \exp_args:NV \__enumext_anskey_env_unknown:nn \l_keys_key_str {#1}
2842 }
2843 \cs_new_protected:Npn \__enumext_anskey_env_unknown:nn #1#2
2844 {
2845     \tl_if_blank:nTF {#2}
2846     {
2847         \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
2848     }
2849     {
2850         \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
2851     }
2852 }

```

(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.

```

2853 \prg_new_protected_conditional:Npnn \__enumext_anskey_env_file_if_writable:n #1 { T, F, TF }
2854 {
2855     \bool_if:NTF \l__enumext_write_anskey_env_bool
2856     {
2857         \file_if_exist:nTF {#1}
2858         {
2859             \bool_if:NTF \l__enumext_anskey_env_overwrite_bool
2860             {
2861                 \msg_warning:nne { enumext } { overwrite-file } {#1}
2862                 \prg_return_true:
2863             }
2864             {
2865                 \msg_warning:nne { enumext } { not-writing } {#1}
2866                 \prg_return_false:
2867             }
2868         }
2869         {
2870             \msg_warning:nne { enumext } { writing-file } {#1}
2871             \prg_return_true:
2872         }
2873     }
2874     { \prg_return_false: }
2875 }

```

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.

```

2876 \cs_new_protected:Npn \__enumext_anskey_env_file_write:nn #1#2
2877 {
2878     \__enumext_anskey_env_file_if_writable:nT {#1}
2879     {
2880         \iow_open:Nn \l__enumext_write_anskey_env_file_iow {#1}
2881         \iow_now:Nn \l__enumext_write_anskey_env_file_iow {#2}
2882         \iow_close:N \l__enumext_write_anskey_env_file_iow

```

```

2883     }
2884   }
2885   \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 `\l__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`.

```

2886 \NewDocumentEnvironment{anskey*}{ o c }
2887 {
2888   \__enumext_anskey_env_safe_outer:
2889   \bool_if:NT \l__enumext_check_answers_bool
2890   {
2891     \tl_if_novalue:nF {#1}
2892     {
2893       \keys_set:nn { enumext / anskey* } {#1}
2894     }
2895     \tl_if_blank:nTF {#2}
2896     {
2897       \msg_error:nn { enumext } { anskey-empty-arg }
2898     }
2899     {
2900       \__enumext_anskey_env_safe_inner:
2901       \__enumext_store_anskey_env:n {#2}
2902     }
2903   }
2904 } { }

```

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

13.31.1 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.

```

2905 \cs_new_protected:Nn \__enumext_anskey_env_safe_outer:
2906 {
2907   \bool_if:NF \l__enumext_store_active_bool
2908   {
2909     \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2910   }
2911   \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2912   {
2913     \msg_error:nnn { enumext } { anskey-env-wrong } { keyans }
2914   }
2915   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2916   {
2917     \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
2918   }
2919   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2920   {
2921     \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
2922   }
2923 }

```

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.

```

2924 \cs_new_protected:Nn \__enumext_anskey_env_safe_inner:
2925 {
2926   \bool_if:NF \l__enumext_item_number_bool
2927   {
2928     \msg_error:nn { enumext } { anskey-unnumber-item }
2929   }
2930   \mode_if_math:T
2931   {
2932     \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
2933   }
2934 }

```

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:N` 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.

```

2935 \cs_new_protected:Npn \__enumext_store_anskey_env:n #1
2936 {
2937   \tl_set:Nn \l__enumext_store_anskey_env_tl {#1}
2938   \RenewDocumentCommand \obeyedline { } { \iow_char:N ^^J }
2939   \tl_replace_all:Nee \l__enumext_store_anskey_env_tl { \obeyedline } { \iow_char:N ^^J }
2940   \__enumext_anskey_env_file_write:VV
2941     \l__enumext_write_anskey_env_file_name_tl \l__enumext_store_anskey_env_tl
2942   \bool_if:NF \l__enumext_anskey_env_force_eol_bool
2943   {
2944     \tl_put_right:Nc \l__enumext_store_anskey_env_tl
2945       {
2946         \c__enumext_anskey_env_hidden_space_str
2947       }
2948   }
2949   \exp_args:N
2950     \__enumext_store_anskey_arg:n
2951     {
2952       \__enumext_scan_tokens:n { \l__enumext_store_anskey_env_tl }
2953     }
2954 }

```

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`.)

13.32 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.

```

2955 \cs_new_protected:Nn \__enumext_execute_after_env:
2956 {
2957   \int_compare:nNnT { \l__enumext_level_int } = { 0 }
2958   {
2959     \tl_if_empty:NF \g__enumext_store_name_tl
2960     {
2961       \__enumext_stop_save_ans_msg:
2962       \__enumext_item_answer_diff:
2963       \__enumext_log_global_vars:
2964       \__enumext_log_answer_vars:
2965       \bool_if:NTF \g__enumext_check_ans_key_bool
2966       {
2967         \__enumext_check_ans_show:
2968       }
2969       { \__enumext_check_ans_log: }
2970     }
2971     \__enumext_reset_global_vars:
2972   }
2973 }

```

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

(End of definition for `__enumext_execute_after_env:`.)

13.33 Common functions for keyans, keyans* and keyanspic

13.33.1 Storing content in prop list

`__enumext_keyans_addto_prop:n`

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

```

2974 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2975 {
2976   \tl_clear:N \__enumext_store_current_label_tl
2977   \int_compare:nNnTF { \__enumext_keyans_pic_level_int } = { 1 }
2978   {
2979     \tl_put_right:Ne \__enumext_store_current_label_tl { \__enumext_label_vi_tl }
2980   }
2981   {
2982     \tl_put_right:Ne \__enumext_store_current_label_tl { \__enumext_label_v_tl }
2983   }

```

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

```

2984   \tl_if_novalue:nF { #1 }
2985   {
2986     \tl_if_empty:NF \__enumext_store_keyans_item_opt_sep_v_tl
2987     {
2988       \tl_put_right:NV \__enumext_store_current_label_tl \__enumext_store_keyans_item_opt_sep_v_tl
2989     }
2990     \tl_put_right:Nn \__enumext_store_current_label_tl { #1 }
2991   }
2992   \__enumext_store_addto_prop:V \__enumext_store_current_label_tl
2993 }

```

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

13.33.2 The save-ref key for keyans, keyans* and keyanspic

The “*internal label and ref*” system for the `keyans`, `keyans*` and `keyanspic` environments has *slight differences* with the one implemented for `\anskey` basically because in this environments the interest is in the current $\langle label \rangle$ for $\backslash item^*$ and $\backslash anspic^*$ with the $\langle contents \rangle$ of the *optional argument*. The mechanism defined here will allow to execute `\ref{<store name: position>}` and will return `1.(A)`.

`__enumext_keyans_store_ref:`

`__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 $\backslash item^*$ and $\backslash anspic^*$ 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.

```

2994 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2995 {
2996   \bool_if:NT \__enumext_store_ref_key_bool
2997   {
2998     \cs_set_protected:Npn \__enumext_tmp:n ##1
2999     {
3000       \tl_set_eq:cc { \__enumext_label_copy_##1_tl } { \__enumext_label_##1_tl }
3001       \tl_reverse:c { \__enumext_label_copy_##1_tl }
3002       \tl_remove_once:cn { \__enumext_label_copy_##1_tl } { . }
3003       \tl_reverse:c { \__enumext_label_copy_##1_tl }
3004     }
3005     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
3006     \__enumext_keyans_store_ref_aux_i:
3007   }
3008 }

```

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`.

```

3009 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
3010 {
3011   \bool_if:NT \g__enumext_starred_bool
3012   {
3013     \tl_set_eq:NN \__enumext_label_copy_i_tl \__enumext_label_copy_vii_tl
3014   }
3015   \int_compare:nNnT { \__enumext_keyans_pic_level_int } = { 1 }
3016   {
3017     \tl_put_right:Ne \__enumext_newlabel_arg_two_tl
3018     { \__enumext_label_copy_i_tl . \__enumext_label_copy_vi_tl }

```

```

3019     }
3020     \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
3021     {
3022         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
3023         { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
3024     }
3025     \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
3026     {
3027         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
3028         { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
3029     }
3030     \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
3031     {
3032         \l__enumext_store_name_tl \c_colon_str
3033         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
3034     }
3035     \__enumext_keyans_store_ref_aux_ii:
3036 }

```

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.

```

3037 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
3038 {
3039     \tl_put_right:Ne \l__enumext_write_aux_file_tl
3040     {
3041         \__enumext_newlabel:nn
3042         { \exp_not:V \l__enumext_newlabel_arg_one_tl }
3043         { \l__enumext_newlabel_arg_two_tl }
3044     }
3045     \l__enumext_write_aux_file_tl
3046 }

```

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

13.33.3 Storing content in sequence

```

\__enumext_keyans_addto_seq:n
\__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.

```

3047 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
3048 {
3049     \tl_clear:N \l__enumext_store_current_label_tl
3050     \int_compare:nNtF { \l__enumext_keyans_pic_level_int } = { 1 }
3051     {
3052         \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
3053     }
3054     {
3055         \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
3056     }
3057     \tl_if_novalue:nF { #1 }
3058     {
3059         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_v_tl
3060         {
3061             \tl_put_right:NV \l__enumext_store_current_label_tl \l__enumext_store_keyans_item_opt_sep_v_tl
3062         }
3063         \tl_put_right:Nn \l__enumext_store_current_label_tl { #1 }
3064     }
3065     \__enumext_keyans_addto_seq_link:
3066 }

```

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.

```

3067 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
3068 {
3069     \bool_lazy_and:nnT

```



```

3070 { \bool_if_p:N \l__enumext_store_ref_key_bool }
3071 { \bool_if_p:N \l__enumext_hyperref_bool }
3072 {
3073   \tl_put_right:Nx \l__enumext_store_current_label_tl
3074   {
3075     \hfill \exp_not:N \hyperlink
3076     {
3077       \exp_not:V \l__enumext_newlabel_arg_one_tl
3078     }
3079     { \exp_not:V \l__enumext_mark_ref_sym_tl }
3080   }
3081 }
3082 \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
3083 \bool_if:NT \l__enumext_check_answers_bool
3084 {
3085   \int_gincr:N \g__enumext_item_anskey_int
3086 }
3087 }

```

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

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

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:

```

```

3088 \cs_new_protected:Npn \__enumext_keyans_save_item_opt:n #1
3089 {
3090   \tl_if_novalue:nF { #1 }
3091   {
3092     \tl_set:Nn \l__enumext_store_current_opt_arg_tl { #1 }
3093   }
3094 }

```

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.

```

3095 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
3096 {
3097   \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
3098   {
3099     \bool_lazy_or:nnT
3100     { \bool_if_p:N \l__enumext_show_answer_bool }
3101     { \bool_if_p:N \l__enumext_show_position_bool }
3102     {
3103       \__enumext_keyans_wrapper_opt_v:n
3104       { \l__enumext_store_current_opt_arg_tl } \c_space_tl
3105     }
3106   }
3107 }

```

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.

```

3108 \cs_new_protected:Nn \__enumext_keyans_show_item_opt_viii:
3109 {
3110   \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
3111   {
3112     \bool_lazy_or:nnT
3113     { \bool_if_p:N \l__enumext_show_answer_bool }
3114     { \bool_if_p:N \l__enumext_show_position_bool }
3115     {
3116       \__enumext_keyans_wrapper_opt_viii:n
3117       { \l__enumext_store_current_opt_arg_tl } \c_space_tl
3118     }
3119   }
3120 }

```

(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`.

```

3121 \cs_new_protected:Nn \__enumext_keyans_pos_mark_set:
3122 {

```

```

3123 \__enumext_label_width_by_box:Nn
3124 \__enumext_mark_sep_tmpa_dim { \__enumext_label_v_tl }
3125 \str_case:Vn \__enumext_align_label_pos_v_str
3126 {
3127   { l }
3128   {
3129     \dim_set:Nn \__enumext_mark_sep_tmpb_dim { \c_zero_dim }
3130   }
3131   { r }
3132   {
3133     \dim_set:Nn \__enumext_mark_sep_tmpb_dim
3134     { \__enumext_labelwidth_v_dim - \__enumext_mark_sep_tmpa_dim }
3135   }
3136   { c }
3137   {
3138     \dim_set:Nn \__enumext_mark_sep_tmpb_dim
3139     { 0.5\__enumext_labelwidth_v_dim - 0.5\__enumext_mark_sep_tmpa_dim }
3140   }
3141 }

```

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

```

3142 \dim_compare:nNt { \__enumext_mark_sym_sep_v_dim } = { \c_zero_dim }
3143 {
3144   \dim_set:Nn \__enumext_mark_sym_sep_v_dim { \__enumext_labelsep_v_dim }
3145 }
3146 \tl_set_eq:NN \__enumext_mark_answer_sym_tl \__enumext_mark_answer_sym_v_tl
3147 \dim_add:Nn \__enumext_mark_sym_sep_v_dim { \__enumext_mark_sep_tmpb_dim }
3148 \str_set_eq:NN \__enumext_mark_position_str \__enumext_mark_position_v_str
3149 }

```

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

```

3150 \cs_new_protected:Nn \__enumext_keyans_show_ans:
3151 {
3152   \bool_lazy_all:nT
3153   {
3154     { \bool_if_p:N \__enumext_show_answer_bool }
3155     { \bool_if_p:N \__enumext_item_wrap_key_bool }
3156   }
3157   {
3158     \__enumext_keyans_pos_mark_set:
3159     \__enumext_print_keyans_box:NN
3160     \__enumext_labelwidth_v_dim \__enumext_mark_sym_sep_v_dim
3161   }
3162 }

```

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.

```

3163 \cs_new_protected:Nn \__enumext_keyans_show_pos:
3164 {
3165   \int_compare:nNtF { \__enumext_keyans_level_int } = { 1 }
3166   {
3167     \int_incr:N \__enumext_show_pos_tmp_int
3168   }
3169   {
3170     \int_zero:N \__enumext_show_pos_tmp_int
3171   }
3172   \bool_lazy_all:nT
3173   {
3174     { \bool_if_p:N \__enumext_show_position_bool }
3175     { \bool_if_p:N \__enumext_item_wrap_key_bool }
3176   }
3177   {
3178     \tl_set:Nn \__enumext_mark_answer_sym_v_tl
3179     {
3180       \group_begin:
3181       \exp_not:N \normalfont
3182       \exp_not:N \footnotesize [ \int_eval:n
3183       {
3184         \prop_count:c { g__enumext_ \__enumext_store_name_tl _prop }
3185         + \__enumext_show_pos_tmp_int
3186       }

```

```

3187         ]
3188         \group_end:
3189     }
3190     \__enumext_keyans_pos_mark_set:
3191     \__enumext_print_keyans_box:NN
3192     \l__enumext_labelwidth_v_dim \l__enumext_mark_sym_sep_v_dim
3193 }
3194 }

```

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

13.34 Redefining `\item` and `\makeLabel` in enumext

Redefining the `\item` command is not as simple as I thought. This command works in conjunction with the `\makeLabel` command so I have to redefine both of them, in addition to this, we will have to use a couple of *global* variables to pass the values from one command to the other.

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

The `\item` and `\item[⟨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`.

```

3195 \cs_new_protected:Npn \__enumext_default_item:n #1
3196 {
3197     \tl_if_novalue:nTF {#1}
3198     {
3199         \bool_if:NT \l__enumext_check_answers_bool
3200         {
3201             \int_gincr:N \g__enumext_item_number_int
3202             \bool_set_true:N \l__enumext_item_number_bool
3203         }
3204         \bool_set_true:c { \l__enumext_wrap_label_ \__enumext_level: _bool }
3205         \__enumext_item_std:w \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
3206     }
3207     {
3208         \bool_set_eq:cc
3209         { \l__enumext_wrap_label_ \__enumext_level: _bool }
3210         { \l__enumext_wrap_label_opt_ \__enumext_level: _bool }
3211         \__enumext_item_std:w [#1] \tl_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
3212     }
3213 }

```

(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`.

```

3214 \cs_new_protected:Npn \__enumext_item_starred_exec:nn #1 #2
3215 {
3216     \tl_if_novalue:nTF {#1}
3217     {
3218         \tl_gset_eq:Nc
3219         \g__enumext_item_symbol_aux_tl { \l__enumext_item_symbol_ \__enumext_level: _tl }
3220     }

```

```

3221     {
3222         \tl_gset:Nn \g__enumext_item_symbol_aux_tl {#1}
3223     }
3224     \tl_if_novalue:nTF {#2}
3225     {
3226         \dim_set_eq:cc
3227         { \__enumext_item_symbol_sep_ \__enumext_level: _dim }
3228         { \__enumext_labelsep_ \__enumext_level: _dim }
3229     }
3230     {
3231         \dim_set:cn { \__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
3232     }
3233     \bool_if:NT \l__enumext_check_answers_bool
3234     {
3235         \int_gincr:N \g__enumext_item_number_int
3236         \bool_set_true:N \l__enumext_item_number_bool
3237     }
3238     \bool_set_true:c { \__enumext_wrap_label_ \__enumext_level: _bool }
3239     \__enumext_item_std:w \tl_use:c { \__enumext_fake_item_indent_ \__enumext_level: _tl }
3240 }

```

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

```

3241 \cs_new_protected:Nn \__enumext_item_starred_exec:
3242 {
3243     \tl_if_empty:cF { \__enumext_item_symbol_ \__enumext_level: _tl }
3244     {
3245         \mode_leave_vertical:
3246         \skip_horizontal:n { -\dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3247         \hbox_overlap_left:n { \g__enumext_item_symbol_aux_tl }
3248         \skip_horizontal:n { \dim_use:c { \__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3249     }
3250 }

```

(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 (§13.39).

```

3251 \cs_new_protected:Nn \__enumext_redefine_item:
3252 {
3253     \RenewDocumentCommand \item { s o o }
3254     {
3255         \bool_if:nTF {##1}
3256         {
3257             \__enumext_item_starred_exec:nn {##2} {##3}
3258         }
3259         { \__enumext_default_item:n {##2} }
3260     }
3261 }

```

(End of definition for `__enumext_redefine_item:.`)

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

```

3262 \cs_new_protected:Nn \__enumext_make_label:
3263 {
3264     \IfDocumentMetadataTF
3265     {
3266         \__enumext_make_label_box:
3267     }
3268     {
3269         \bool_if:NTF \l__enumext_mode_box_bool
3270         {
3271             \__enumext_make_label_box:
3272         }
3273         {
3274             \__enumext_make_label_std:
3275         }
3276     }
3277 }

```

Standard definition when `\DocumentMetadata` is not active.

```

3278 \cs_new_protected:Nn \__enumext_make_label_std:
3279 {
3280   \RenewDocumentCommand \make_label { m }
3281   {
3282     \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
3283     \__enumext_item_starred_exec:
3284     \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3285     \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3286     {
3287       \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3288     }
3289     { ##1 }
3290     \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
3291     \tl_gclear:N \g__enumext_item_symbol_aux_tl
3292   }
3293 }

```

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`.

```

3294 \cs_new_protected:Nn \__enumext_make_label_box:
3295 {
3296   \RenewDocumentCommand \make_label { m }
3297   {
3298     \strut\smash
3299     {
3300       \makebox
3301       [ \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim } ]
3302       [ \str_use:c { l__enumext_align_label_pos_ \__enumext_level: _str } ]
3303       {
3304         \__enumext_item_starred_exec:
3305         \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3306         \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3307         {
3308           \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3309         }
3310         { ##1 }
3311         \tl_gclear:N \g__enumext_item_symbol_aux_tl
3312       }
3313     } % close smash
3314   }
3315 }

```

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

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

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

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

```

item-pos*
3316 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
3317 {
3318   \keys_define:nn { enumext / #1 }
3319   {
3320     item-sym* .tl_set:c = { l__enumext_item_symbol_#2_tl },
3321     item-sym* .value_required:n = true,
3322     item-sym* .initial:n = {\textborn},
3323     item-pos* .dim_set:c = { l__enumext_item_symbol_sep_#2_dim },
3324     item-pos* .value_required:n = true,
3325   }
3326 }
3327 \clist_map_inline:nn
3328 {
3329   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
3330 }
3331 { \__enumext_tmp:nn #1 }

```

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

13.36 Handling unknown keys

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

13.36.1 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
3332 \cs_set_protected:Npn \__enumext_tmp:n #1
3333 {
3334   \keys_define:nn { enumext / #1 }
3335   {
3336     unknown .code:n = {
3337       \tl_set:Nn \l__enumext_envir_name_tl {#1}
3338       \__enumext_keyans_unknown_keys:n {#1}
3339     },
3340   }
3341 }
3342 \clist_map_inline:nn { keyans, keyans*, keyanspic } { \__enumext_tmp:n {#1} }
```

Internal functions for handling `unknown` key.

```

3343 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:n #1
3344 {
3345   \exp_args:NV \__enumext_keyans_unknown_keys:nn \l_keys_key_str {#1}
3346 }
3347 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:nn #1#2
3348 {
3349   \tl_if_blank:nTF {#2}
3350   {
3351     \msg_error:nne { enumext } { keyans-unknown-key } {#1}
3352   }
3353   {
3354     \msg_error:nnee { enumext } { keyans-unknown-key-value } {#1} {#2}
3355   }
3356 }
```

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

13.36.2 Handling unknown keys for enumext*

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

```

unknown
\__enumext_starred_unknown_keys:n
\__enumext_starred_unknown_keys:nn
3357 \keys_define:nn { enumext / enumext* }
3358 {
3359   unknown .code:n = { \__enumext_starred_unknown_keys:n {#1} },
3360 }
```

Internal functions for handling `unknown` key.

```

3361 \cs_new_protected:Npn \__enumext_starred_unknown_keys:n #1
3362 {
3363   \exp_args:NV \__enumext_starred_unknown_keys:nn \l_keys_key_str {#1}
3364 }
3365 \cs_new_protected:Npn \__enumext_starred_unknown_keys:nn #1#2
3366 {
3367   \tl_if_blank:nTF {#2}
3368   {
3369     \msg_error:nnn { enumext } { starred-unknown-key } {#1}
3370   }
3371   {
3372     \msg_error:nnnn { enumext } { starred-unknown-key-value } {#1} {#2}
3373   }
3374 }
```

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

13.36.3 Handling unknown keys for enumext

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

```

3375 \cs_set_protected:Npn \__enumext_tmp:n #1
3376 {
3377   \keys_define:nn { enumext / #1 }
3378   {
3379     unknown .code:n = { \__enumext_standar_unknown_keys:n {##1} },
3380   }
3381 }
3382 \clist_map_inline:nn { level-1,level-2,level-3,level-4 } { \__enumext_tmp:n {#1} }

```

Internal functions for handling `unknown` key.

```

3383 \cs_new_protected:Npn \__enumext_standar_unknown_keys:n #1
3384 {
3385   \exp_args:NV \__enumext_standar_unknown_keys:nn \l_keys_key_str {#1}
3386 }
3387 \cs_new_protected:Npn \__enumext_standar_unknown_keys:nn #1#2
3388 {
3389   \tl_if_blank:nTF {#2}
3390   {
3391     \msg_error:nnn { enumext } { standar-unknown-key } {#1}
3392   }
3393   {
3394     \msg_error:nnnn { enumext } { standar-unknown-key-value } {#1} {#2}
3395   }
3396 }

```

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

13.37 Redefining `\item` and `\makeLabel` in keyans

The `\item` and `\item[⟨custom⟩]` commands work in the usual way in `keyans`, but the `\item*` and `\item*[⟨content⟩]` commands *store* the current `⟨label⟩` next to the `⟨content⟩` if it is present in the *sequence* and *prop list* defined by `save-ans` key.

`__enumext_keyans_default_item:n`

The function `__enumext_keyans_default_item:n` executes the original behavior of the `\item` along with the keys `wrap-label`, `wrap-label*` and `itemindent`.

```

3397 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
3398 {
3399   \tl_if_novalue:nTF { #1 }
3400   {
3401     \bool_set_true:N \l__enumext_wrap_label_v_bool
3402     \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
3403   }
3404   {
3405     \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
3406     \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
3407   }
3408 }

```

(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.

```

3409 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
3410 {
3411   \__enumext_keyans_save_item_opt:n { #1 }
3412   \bool_set_true:N \l__enumext_wrap_label_v_bool
3413   \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
3414   \__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.

```

3415   \__enumext_keyans_addto_prop:n { #1 }
3416   \__enumext_keyans_store_ref:
3417   \__enumext_keyans_addto_seq:n { #1 }
3418   \int_gincr:N \g__enumext_check_starred_cmd_int
3419 }

```


(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 (§13.38).

```

3420 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
3421 {
3422   \RenewDocumentCommand \item { s o }
3423   {
3424     \bool_if:nTF {##1}
3425     {
3426       \bool_set_true:N \l__enumext_item_wrap_key_bool % wrap-ans*
3427       \peek_remove_spaces:n
3428       {
3429         \__enumext_keyans_starred_item:n {##2}
3430       }
3431     }
3432     {
3433       \bool_set_false:N \l__enumext_item_wrap_key_bool
3434       \__enumext_keyans_default_item:n {##2}
3435     }
3436   }
3437 }

```

(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 (§13.38).

```

3438 \cs_new_protected:Nn \__enumext_keyans_make_label:
3439 {
3440   \IfDocumentMetadataTF
3441   {
3442     \__enumext_keyans_make_label_box:
3443   }
3444   {
3445     \bool_if:NTF \l__enumext_mode_box_bool
3446     {
3447       \__enumext_keyans_make_label_box:
3448     }
3449     {
3450       \__enumext_keyans_make_label_std:
3451     }
3452   }
3453 }

```

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

```

3454 \cs_new_protected:Npn \__enumext_keyans_wrapper_label:n #1
3455 {
3456   \bool_lazy_all:nT
3457   {
3458     { \bool_if_p:N \l__enumext_wrap_label_v_bool }
3459     { \bool_if_p:N \l__enumext_show_answer_bool }
3460     { \bool_if_p:N \l__enumext_item_wrap_key_bool }
3461     { \cs_if_exist_p:N \__enumext_keyans_wrapper_item_v:n }
3462   }
3463   {
3464     \cs_set_eq:NN \__enumext_wrapper_label_v:n \__enumext_keyans_wrapper_item_v:n
3465   }
3466   \bool_if:NTF \l__enumext_wrap_label_v_bool
3467   {
3468     \__enumext_wrapper_label_v:n { #1 }
3469   }
3470   { #1 }
3471 }

```

Standard definition when `\DocumentMetadata` is not active.

```

3472 \cs_new_protected:Nn \__enumext_keyans_make_label_std:
3473 {
3474   \RenewDocumentCommand \makeLabel { m }
3475   {
3476     \tl_use:N \l__enumext_label_fill_left_v_tl
3477     \__enumext_keyans_show_ans:
3478     \__enumext_keyans_show_pos:
3479     \tl_use:N \l__enumext_label_font_style_v_tl
3480     \__enumext_keyans_wrapper_label:n { ##1 }
3481     \tl_use:N \l__enumext_label_fill_right_v_tl
3482   }
3483 }

```

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

```

3484 \cs_new_protected:Nn \__enumext_keyans_make_label_box:
3485 {
3486   \RenewDocumentCommand \makeLabel { m }
3487   {
3488     \strut\smash
3489     {
3490       \makebox[ \l__enumext_labelwidth_v_dim ][ \l__enumext_align_label_pos_v_str ]
3491       {
3492         \__enumext_keyans_show_ans:
3493         \__enumext_keyans_show_pos:
3494         \tl_use:N \l__enumext_label_font_style_v_tl
3495         \__enumext_keyans_wrapper_label:n { ##1 }
3496       }
3497     }
3498   }
3499 }

```

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

13.38 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.

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

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

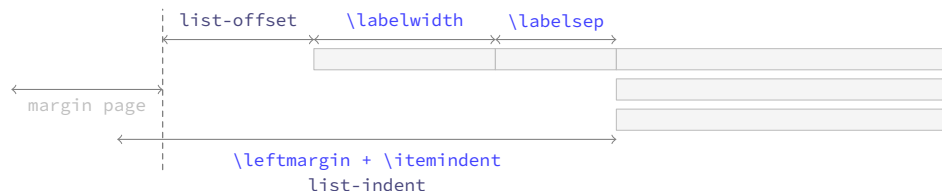


Figure 9: Representation of standard horizontal lengths in `list` environment.

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

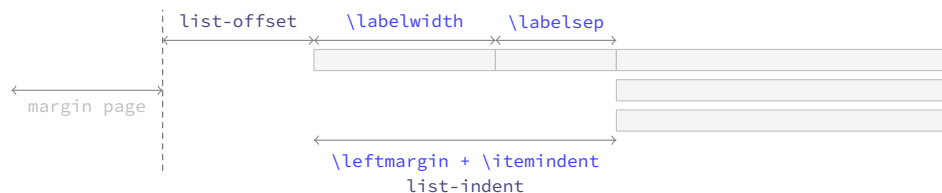


Figure 10: Representation of horizontal lengths concept in list in `enumext`.

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

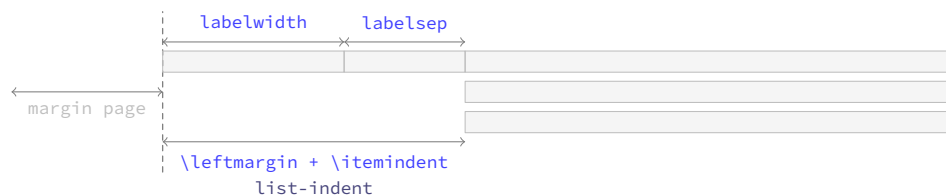


Figure 11: Default horizontal lengths in enumext.

```
\__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`.

```
3500 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
3501 {
3502   \dim_compare:nNt { #1 } < { \c_zero_dim }
3503   {
3504     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
3505     \dim_set:Nn #1 { \dim_abs:n { #1 } }
3506   }
3507   \dim_compare:nNt { #2 } < { \c_zero_dim }
3508   {
3509     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
3510     \dim_set:Nn #2 { \dim_abs:n { #2 } }
3511   }

```

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

```
3512   \bool_if:NF #7 { \dim_set:Nn #4 { #1 + #2 } }
```

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

```
3513   \dim_compare:nNtF { #4 } < { \c_zero_dim }
3514   {
3515     \dim_set:Nn #6 { #1 + #2 - #4 }
3516     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3517   }
3518   {
3519     \dim_compare:nNt { #4 } = { #1 + #2 }
3520     { \dim_set:Nn #6 { \c_zero_dim } }
3521     \dim_compare:nNt { #4 } < { #1 + #2 }
3522     { \dim_set:Nn #6 { #1 + #2 - #4 } }
3523     \dim_compare:nNt { #4 } > { #1 + #2 }
3524     {
3525       \dim_set:Nn #6 { -#1 - #2 + #4 }
3526       \dim_set:Nn #6 { #6*-1 }
3527     }
3528     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3529   }
3530 }
3531 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { ccccccc }
```

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

13.38.2 Setting second argument of the lists

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

```
3532 \cs_set_protected:Npn \__enumext_tmp:n #1
3533 {
3534   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3535   {
3536     \__enumext_calc_hspace:ccccccc
3537     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3538     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3539     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3540     { \__enumext_leftmargin_tmp_#1_bool }
```

```

3541 \clist_map_inline:nn
3542 { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3543 { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
3544 \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3545 { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
3546 \usecounter { enumX#1 }
3547 \str_if_eq:nnTF {#1} { v }
3548 {
3549   \__enumext_keyans_redefine_item:
3550   \__enumext_keyans_make_label:
3551   \__enumext_keyans_ref:
3552   \__enumext_keyans_fake_item_indent:
3553   \bool_if:cT { \__enumext_show_length_#1_bool }
3554   {
3555     \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3556   }
3557 }
3558 {
3559   \__enumext_redefine_item:
3560   \__enumext_make_label:
3561   \__enumext_standar_ref:
3562   \__enumext_fake_item_indent:
3563   \bool_if:cT { \__enumext_show_length_#1_bool }
3564   {
3565     \msg_term:nnne { enumext } { list-lengths } {#1}
3566     { \int_use:N \__enumext_level_int }
3567   }
3568 }
3569 }
3570 }
3571 \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 `0pt`. 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 `\parskip` locally.

```

3572 \cs_set_protected:Npn \__enumext_tmp:n #1
3573 {
3574   \cs_new_protected:cpn { \__enumext_list_arg_two_#1: }
3575   {
3576     \bool_set_true:c { \__enumext_leftmargin_tmp_#1_bool }
3577     \dim_zero:c { \__enumext_leftmargin_tmp_#1_dim }
3578     \__enumext_calc_hspace:cccccc
3579     { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3580     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3581     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3582     { \__enumext_leftmargin_tmp_#1_bool }
3583     \clist_map_inline:nn
3584     { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3585     { \dim_set_eq:cc {####1} { \__enumext_####1_#1_dim } }
3586     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3587     { \skip_set_eq:cc {####1} { \__enumext_####1_#1_skip } }
3588     \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
3589     \skip_zero:N \partopsep
3590     \usecounter { enumX#1 }
3591     \__enumext_starred_ref:
3592     \str_if_eq:nnTF {#1} { vii }
3593     {
3594       \__enumext_fake_item_indent_vii:
3595       \bool_if:cT { \__enumext_show_length_vii_bool }
3596       { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3597     }
3598     {
3599       \__enumext_fake_item_indent_viii:
3600       \bool_if:cT { \__enumext_show_length_#1_bool }
3601       { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3602     }
3603   }
3604 }

```

```
3605 \clist_map_inline:nn { vii, viii } { \__enumext_tmp:n {#1} }
```

(End of definition for __enumext_list_arg_two_vii: and __enumext_list_arg_two_viii:.)

13.39 The environment enumext

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

```
3606 \cs_new_protected:Nn \__enumext_safe_exec:
3607 {
3608   \__enumext_is_not_nested:
3609   \__enumext_internal_mini_page:
3610   \int_incr:N \l__enumext_level_int
3611   \int_compare:nNt { \l__enumext_level_int } > { 4 }
3612   { \msg_fatal:nn { enumext } { list-too-deep } }
3613   \bool_set_true:N \l__enumext_standar_bool
3614   \bool_set_false:N \l__enumext_starred_bool
3615   \__enumext_is_on_first_level:
3616 }
```

(End of definition for __enumext_safe_exec:.)

__enumext_parse_keys:n The __enumext_parse_store_keys:n function first we will clear the variable \l__enumext_series_str used by the key `series` and then we check if we are at the “first level”, if so we process the `(keys)` and then execute the function __enumext_parse_series:n used by the key `series` and call the function __enumext_nested_base_line_fix: used by the key `base-fix`, otherwise we will pass the `(keys)` to the inner levels of the environment then we execute the function __enumext_store_active_keys:n and reprocess the `(keys)` to pass them to the `sequence` if the key `save-key` is not active.

```
3617 \cs_new_protected:Npn \__enumext_parse_keys:n #1
3618 {
3619   \tl_if_novalue:nF {#1}
3620   {
3621     \str_clear:N \l__enumext_series_str
3622     \int_compare:nNtF { \l__enumext_level_int } = { 1 }
3623     {
3624       \keys_set:nn { enumext / level-1 } {#1}
3625       \__enumext_parse_series:n {#1}
3626       \__enumext_nested_base_line_fix:
3627     }
3628     {
3629       \exp_args:Ne \keys_set:nn
3630       { enumext / level-\int_use:N \l__enumext_level_int } {#1}
3631     }
3632     \__enumext_store_active_keys:n {#1}
3633   }
3634 }
```

(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*`.

```
3635 \cs_new_protected:Nn \__enumext_start_store_level:
3636 {
3637   \bool_lazy_all:nT
3638   {
3639     { \bool_if_p:N \l__enumext_store_active_bool }
3640     { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3641     { \bool_if_p:N \g__enumext_standar_bool }
3642   }
3643   {
3644     \int_compare:nNt { \l__enumext_level_int } > { 1 }
3645     {
3646       \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3647       \__enumext_store_level_open:
3648     }
3649   }
```

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

```

3650     \bool_lazy_all:nT
3651     {
3652       { \bool_if_p:N \__enumext_store_active_bool }
3653       { \bool_not_p:n { \__enumext_keyans_env_bool } }
3654       { \int_compare_p:nNn { \__enumext_level_h_int } = { 1 } }
3655     }
3656     {
3657       \int_compare:nNnT { \__enumext_level_int } > { 0 }
3658       {
3659         \bool_set_true:c { \__enumext_store_upper_level_ \__enumext_level: _bool }
3660         \__enumext_store_level_open:
3661       }
3662     }
3663   }

```

(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*`.

```

3664 \cs_new_protected:Nn \__enumext_stop_store_level:
3665 {
3666   \bool_if:cT { \__enumext_store_upper_level_ \__enumext_level: _bool }
3667   {
3668     \__enumext_store_level_close:
3669   }
3670 }

```

(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.

```

3671 \cs_new_protected:Nn \__enumext_multicols_start:
3672 {
3673   \int_compare:nNnT
3674   { \int_use:c { \__enumext_columns_ \__enumext_level: _int } } > { 1 }
3675   {
3676     \dim_compare:nNnT
3677     { \dim_use:c { \__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3678     {
3679       \dim_set:cn { \__enumext_columns_sep_ \__enumext_level: _dim }
3680       {
3681         ( \dim_use:c { \__enumext_labelwidth_ \__enumext_level: _dim }
3682         + \dim_use:c { \__enumext_labelsep_ \__enumext_level: _dim }
3683         ) / \int_use:c { \__enumext_columns_ \__enumext_level: _int }
3684         - \dim_use:c { \__enumext_listoffset_ \__enumext_level: _dim }
3685       }
3686     }
3687     \dim_set_eq:Nc \columnsep { \__enumext_columns_sep_ \__enumext_level: _dim }
3688     \int_compare:nNnT { \__enumext_level_int } > { 1 }
3689     {
3690       \dim_zero:N \columnseprule
3691     }
3692   }

```

We will calculate the *vertical spacing* settings for the `multicols` environment using the function `__enumext_multi_advspace:`, apply our “*vertical adjust spacing*”, then start the `multicols` environment.

```

3692     \bool_if:cF { \__enumext_minipage_active_ \__enumext_level: _bool }
3693     {
3694       \skip_zero:N \multicolsep
3695       \__enumext_multi_advspace:
3696     }
3697     \raggedcolumns
3698     \begin{multicols}{ \int_use:c { \__enumext_columns_ \__enumext_level: _int } }
3699   }
3700 }

```

(End of definition for `__enumext_multicols_start:`.)

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

```

3701 \cs_new_protected:Nn \__enumext_multicols_stop:
3702 {
3703   \int_compare:nNtF
3704     { \int_use:c { \__enumext_columns_ \__enumext_level: _int } } > { 1 }
3705   {
3706     \__enumext_stop_list:
3707     \__enumext_stop_store_level:
3708     \end{multicols}
3709     \__enumext_unskip_unkern:
3710     \__enumext_unskip_unkern:
3711     \par\addvspace{ \skip_use:c { \__enumext_multicols_below_ \__enumext_level: _skip } }
3712   }
3713   {
3714     \__enumext_stop_list:
3715     \__enumext_stop_store_level:
3716   }
3717 }

```

(End of definition for `__enumext_multicols_stop:`)

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

```

3718 \cs_new_protected:Nn \__enumext_before_list:
3719 {
3720   \__enumext_vspace_above:
3721   \__enumext_before_args_exec:
3722   \__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.

```

3723 \dim_compare:nNtF
3724   { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3725   {
3726     \dim_set:cn { \l__enumext_minipage_left_ \__enumext_level: _dim }
3727     {
3728       \linewidth
3729       - \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim }
3730       - \dim_use:c { \l__enumext_minipage_hsep_ \__enumext_level: _dim }
3731     }

```

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

```

3732   \bool_set_true:c { \l__enumext_minipage_active_ \__enumext_level: _bool }
3733   \int_gincr:N \g__enumext_minipage_stat_int
3734   \__enumext_minipage_add_space:
3735   \noindent
3736   \__enumext_mini_page{ \dim_use:c { \l__enumext_minipage_left_ \__enumext_level: _dim } }
3737 }
3738 \__enumext_multicols_start:
3739 }

```

(End of definition for `__enumext_before_list:`)

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

```

3740 \cs_new_protected:Nn \__enumext_second_part:

```



```

3741 {
3742   \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
3743   {
3744     \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
3745     {
3746       \msg_warning:nn { enumext } { missing-miniright }
3747       \miniright
3748     }
3749     \int_gzero:N \g__enumext_minipage_stat_int
3750     \__enumext_unskip_unkern: % remove topsep + [partopsep]
3751     \end__enumext_mini_page
3752   }
3753   {
3754     \__enumext_multicols_stop:
3755   }

```

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.

```

3756   \__enumext_after_stop_list:
3757   \__enumext_check_ans_key_hook:
3758   \__enumext_vspace_below:
3759   \bool_set_false:N \l__enumext_standar_bool
3760   \__enumext_resume_save_counter:
3761 }

```

(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.

```

3762 \cs_new_protected:Nn \__enumext_set_item_width:
3763 {
3764   \dim_set:Nn \itemwidth { \linewidth }
3765   \dim_compare:nT
3766   {
3767     \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim } != \c_zero_dim
3768   }
3769   {
3770     \dim_sub:Nn \itemwidth
3771     {
3772       \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3773     }
3774   }
3775 }

```

(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.

```

3776 \cs_new_protected:Nn \__enumext_start_counter:
3777 {
3778   \setcounter { enumX \__enumext_level: }
3779   {
3780     \int_eval:n { \int_use:c { l__enumext_start_ \__enumext_level: _int } - 1 }
3781   }
3782 }

```

(End of definition for `__enumext_start_counter:`)

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

```

3783 \NewDocumentEnvironment{enumext}{0} {
3784 {
3785   \__enumext_safe_exec:
3786   \__enumext_parse_keys:n {#1}
3787   \__enumext_before_list:
3788   \__enumext_start_store_level:

```

```

3789   \__enumext_start_list:nn
3790   { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
3791   {
3792     \use:c { __enumext_list_arg_two_ \__enumext_level: : }
3793     \__enumext_before_keys_exec:
3794   }
3795   \__enumext_start_counter:
3796   \__enumext_set_item_width:
3797   \__enumext_after_args_exec:
3798 }
3799 {
3800   \__enumext_second_part:
3801 }

```

(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`.

```

3802 \__enumext_after_env:nn {enumext}
3803 {
3804   \__enumext_execute_after_env:
3805 }

```

13.40 The environment `keyans`

The environment `keyans` also based on lists. The main differences with the `enumext` environment are the *nesting* and the way the *answers* (choice) will be stored and checked, this environment is intended exclusively for “multiple choice questions”.

The `keyans` environment will only be available if the `save-ans` key is active and can only be used at the “first level” within the `enumext` environment. We do not want the environment to be nested, so we will set a maximum at this point. If the conditions are not met, an error message will be returned.

```

3806 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3807 {
3808   \bool_if:NF \l__enumext_store_active_bool
3809   {
3810     \msg_error:nnnn { enumext } { wrong-place } { keyans } { save-ans }
3811   }
3812   \int_incr:N \l__enumext_keyans_level_int
3813   \bool_set_true:N \l__enumext_keyans_env_bool
3814   \__enumext_keyans_name_and_start:
3815   % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3816   \bool_set_false:N \l__enumext_store_active_bool
3817   \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
3818   {
3819     \msg_error:nn { enumext } { keyans-nested }
3820   }
3821   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3822   {
3823     \msg_error:nn { enumext } { keyans-wrong-level }
3824   }
3825 }

```

(End of definition for `__enumext_keyans_safe_exec:.`)

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

```

3826 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3827 {
3828   \keys_set:nn { enumext / keyans } {#1}
3829 }

```

(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:
3830 \cs_new_protected:Nn \__enumext_before_list_v:
3831 {
3832   \__enumext_vspace_above_v:
3833   \__enumext_before_args_exec_v:
3834   \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3835   {
3836     \dim_set:Nn \l__enumext_minipage_left_v_dim
3837     {

```

```

3838         \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3839     }
3840     \bool_set_true:N \l__enumext_minipage_active_v_bool
3841     \int_gincr:N \g__enumext_minipage_stat_int
3842     \__enumext_keyans_minipage_add_space:
3843     \__enumext_mini_page{ \l__enumext_minipage_left_v_dim }
3844 }
3845 \__enumext_keyans_multicols_start:
3846 }
3847 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3848 {
3849     \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3850     {
3851         \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3852         {
3853             \dim_set:Nn \l__enumext_columns_sep_v_dim
3854             {
3855                 (
3856                     \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3857                 ) / \l__enumext_columns_v_int
3858                 - \l__enumext_listoffset_v_dim
3859             }
3860         }
3861         \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim
3862         \dim_zero:N \columnseprule % no rule here
3863         \bool_if:NF \l__enumext_minipage_active_v_bool
3864         {
3865             \skip_zero:N \multicolsep
3866             \__enumext_keyans_multi_addvspace:
3867         }
3868         \raggedcolumns
3869         \begin{multicols}{\l__enumext_columns_v_int}
3870     }
3871 }
3872 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3873 {
3874     \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
3875     {
3876         \__enumext_stop_list:
3877         \end{multicols}
3878         \__enumext_unskip_unkern:
3879         \__enumext_unskip_unkern:
3880         \par\addvspace{ \l__enumext_multicols_below_v_skip }
3881     }
3882     {
3883         \__enumext_stop_list:
3884     }
3885 }
3886 \cs_new_protected:Nn \__enumext_second_part_v:
3887 {
3888     \bool_if:NTF \l__enumext_minipage_active_v_bool
3889     {
3890         \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3891         {
3892             \msg_warning:nn { enumext } { missing-miniright }
3893             \miniright
3894         }
3895         \int_gzero:N \g__enumext_minipage_stat_int
3896         \__enumext_unskip_unkern: % remove \topsep + [\partopsep]
3897         \end__enumext_mini_page
3898         \par\addvspace{ \l__enumext_minipage_after_skip }
3899     }
3900     {
3901         \__enumext_keyans_multicols_stop:
3902     }
3903     \bool_set_false:N \l__enumext_keyans_env_bool
3904     \__enumext_after_stop_list_v:
3905     \__enumext_vspace_below_v:
3906 }

```

(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.

```

3907 \cs_new_protected:Nn \__enumext_keyans_set_item_width:
3908 {
3909   \dim_set:Nn \itemwidth { \linewidth }
3910   \dim_compare:nT
3911     {
3912       \__enumext_listoffset_v_dim != \c_zero_dim
3913     }
3914   {
3915     \dim_sub:Nn \itemwidth { \__enumext_listoffset_v_dim }
3916   }
3917 }

```

(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}`.

```

3918 \cs_new_protected:Nn \__enumext_keyans_start_counter:
3919 {
3920   \setcounter { enumXv } { \int_eval:n { \int_use:c { \__enumext_start_v_int } - 1 } }
3921 }

```

(End of definition for `__enumext_keyans_start_counter:`.)

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

```

3922 \NewDocumentEnvironment{keyans}{0}{}
3923 {
3924   \__enumext_keyans_safe_exec:
3925   \__enumext_keyans_parse_keys:n {#1}
3926   \__enumext_before_list_v:
3927   \__enumext_start_list:nn
3928     { \tl_use:N \__enumext_label_v_tl }
3929   {
3930     \__enumext_list_arg_two_v:
3931     \__enumext_before_keys_exec_v:
3932   }
3933   \__enumext_keyans_start_counter:
3934   \__enumext_keyans_set_item_width:
3935   \__enumext_after_args_exec_v:
3936 }
3937 {
3938   \__enumext_check_starred_cmd:n { item }
3939   \__enumext_second_part_v:
3940 }

```

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

13.41 Tagging PDF support for non-standart list environments

The \LaTeX 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](#).

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

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

```

start-list-tags
stop-start-tags
stop-list-tags
\__enumext_start_list_tag:n
\__enumext_stop_start_list_tag:
\__enumext_stop_list_tag:n
3941 \socket_new:nn {tagsupport/\__enumext/starred}{1}
3942 \socket_new_plugin:nnn {tagsupport/\__enumext/starred} {start-list-tags}
3943 {
3944   \tag_resume:n {#1}
3945   \tag_mc_end_push:
3946     \tag_struct_begin:n {tag=LI}
3947     \tag_struct_begin:n {tag=Lbl}
3948     \tag_mc_begin:n {tag=Lbl}
3949 }

```

```

3950 \socket_new_plug:nnn {tagsupport/__enumext/starred} {stop-start-tags}
3951 {
3952     \tag_mc_end:
3953     \tag_struct_end:n {tag=Lbl}
3954     \tag_struct_begin:n {tag=LBody}
3955     \tag_struct_begin:n {tag=text-unit}
3956     \tag_struct_begin:n {tag=text}
3957 }
3958 \socket_new_plug:nnn {tagsupport/__enumext/starred} {stop-list-tags}
3959 {
3960     \tag_struct_end:n {tag=text}
3961     \tag_struct_end:n {tag=text-unit}
3962     \tag_struct_end:n {tag=LBody}
3963     \tag_struct_end:n {tag=LI}
3964     \tag_mc_begin_pop:n {}
3965     \tag_suspend:n {#1}
3966 }

```

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

```

3967 \cs_new_protected_nopar:Npn __enumext_start_list_tag:n #1
3968 {
3969     \IfDocumentMetadataT
3970     {
3971         \socket_assign_plug:nn {tagsupport/__enumext/starred} {start-list-tags}
3972         \socket_use:nn {tagsupport/__enumext/starred} {#1}
3973     }
3974 }
3975 \cs_new_protected_nopar:Nn __enumext_stop_start_list_tag:
3976 {
3977     \IfDocumentMetadataT
3978     {
3979         \socket_assign_plug:nn {tagsupport/__enumext/starred} {stop-start-tags}
3980         \socket_use:nn {tagsupport/__enumext/starred} { }
3981     }
3982 }
3983 \cs_new_protected_nopar:Npn __enumext_stop_list_tag:n #1
3984 {
3985     \IfDocumentMetadataT
3986     {
3987         \socket_assign_plug:nn {tagsupport/__enumext/starred} {stop-list-tags}
3988         \socket_use:nn {tagsupport/__enumext/starred} {#1}
3989     }
3990 }

```

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

13.41.2 Socket for tagging support in keyanspic

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

```

start-list-tags
stop-start-tags
stop-list-tags
__enumext_anspic_start_list_tag:
__enumext_anspic_stop_start_list_tag:
__enumext_anspic_stop_list_tag:

```

```

3991 \socket_new:nn {tagsupport/__enumext/keyanspic}{ 0 }
3992 \socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {start-list-tags}
3993 {
3994     \tag_resume:n {keyanspic}
3995     \tag_mc_end_push:
3996     \tag_struct_begin:n {tag=LI}
3997     \tag_struct_begin:n {tag=Lbl}
3998     \tag_mc_begin:n {tag=Lbl}
3999 }
4000 \socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {stop-start-tags}
4001 {
4002     \tag_mc_end:
4003     \tag_struct_end:n {tag=Lbl}
4004     \tag_struct_begin:n {tag=LBody}
4005     \tag_struct_begin:n {tag=text-unit}
4006     \tag_struct_begin:n {tag=text}
4007     \tag_mc_begin:n {tag=text}
4008 }
4009 \socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {stop-list-tags}
4010 {
4011     \tag_mc_end:
4012     \tag_struct_end:n {tag=text}
4013     \tag_struct_end:n {tag=text-unit}
4014     \tag_struct_end:n {tag=LBody}

```

```

4015     \tag_struct_end:n {tag=LI}
4016     \tag_mc_begin_pop:n {}
4017     \tag_suspend:n {keyanspic}
4018 }

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

4019 \cs_new_protected_nopar:Nn \__enumext_anspic_start_list_tag:
4020 {
4021     \IfDocumentMetadataT
4022     {
4023         \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {start-list-tags}
4024         \socket_use:n {tagsupport/__enumext/keyanspic}
4025     }
4026 }
4027 \cs_new_protected_nopar:Nn \__enumext_anspic_stop_start_list_tag:
4028 {
4029     \IfDocumentMetadataT
4030     {
4031         \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {stop-start-tags}
4032         \socket_use:n {tagsupport/__enumext/keyanspic}
4033     }
4034 }
4035 \cs_new_protected_nopar:Nn \__enumext_anspic_stop_list_tag:
4036 {
4037     \IfDocumentMetadataT
4038     {
4039         \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {stop-list-tags}
4040         \socket_use:n {tagsupport/__enumext/keyanspic}
4041     }
4042 }

```

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

13.42 The environment `keyanspic` and `\anspic`

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

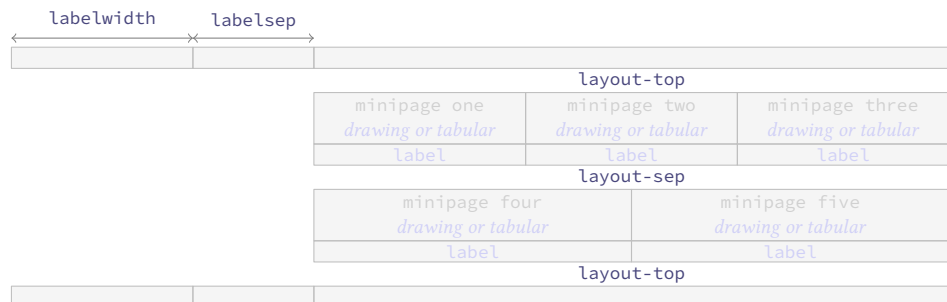


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

In order for the `keyanspic` environment and the `\anspic` command to work correctly, we need to set and export some variables in the first part of the environment definition and pass them to `\anspic` which is executed in the second part of the environment. This implementation is adapted from the answer given by Enrico Gregorio (@egreg) in [How to process the body of an environment and divide it by a \macro?](#).

13.42.1 The environment `keyanspic`

First we define the key that allows us to process the position of the $\langle label \rangle$ 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 $\{n^{\circ} upper, n^{\circ} lower\}$ 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`.

```

mark-ans 4043 \keys_define:nn { enumext / keyanspic }
mark-pos
mark-sep
save-sep 4044 {
wrap-opt 4045     label-pos .choice:,
wrap-ans* 4046     label-pos / above .code:n =
show-ans 4047         \bool_set_true:N \l__enumext_anspic_label_above_bool
show-pos 4048         \str_set:Nn \l__enumext_anspic_mini_pos_str { t },

```

```

4049     label-pos / below .code:n =
4050         \bool_set_false:N \l__enumext_anspic_label_above_bool
4051         \str_set:Nn \l__enumext_anspic_mini_pos_str { b },
4052     label-pos / unknown .code:n =
4053         \msg_error:nnee { enumext } { unknown-choice }
4054         { label-pos } { above,~ below } { \exp_not:n {#1} },
4055     label-pos .initial:n = below,
4056     label-pos .value_required:n = true,
4057     label-sep .skip_set:N = \l__enumext_anspic_label_sep_skip,
4058     label-sep .value_required:n = true,
4059     layout-sty .tl_set:N = \l__enumext_anspic_layout_style_tl,
4060     layout-sty .value_required:n = true,
4061     layout-sep .code:n = \keys_set:nn { enumext / keyans } { parsep = #1 },
4062     layout-sep .value_required:n = true,
4063     layout-top .code:n = \keys_set:nn { enumext / keyans } { topsep = #1 },
4064     layout-top .value_required:n = true,
4065     mark-ans .code:n = \keys_set:nn { enumext / keyans } { mark-ans = #1 },
4066     mark-ans .value_required:n = true,
4067     mark-pos .code:n = \keys_set:nn { enumext / keyans } { mark-pos = #1 },
4068     mark-pos .value_required:n = true,
4069     mark-sep .code:n = \keys_set:nn { enumext / keyans } { mark-sep = #1 },
4070     mark-sep .value_required:n = true,
4071     save-sep .code:n = \keys_set:nn { enumext / keyans } { save-sep = #1 },
4072     save-sep .value_required:n = true,
4073     wrap-opt .code:n = \keys_set:nn { enumext / keyans } { wrap-opt = #1 },
4074     wrap-opt .value_required:n = true,
4075     wrap-ans* .code:n = \keys_set:nn { enumext / keyans } { wrap-ans* = #1 },
4076     wrap-ans* .value_required:n = true,
4077     show-ans .code:n = \keys_set:nn { enumext / keyans } { show-ans = #1 },
4078     show-ans .value_required:n = true,
4079     show-pos .code:n = \keys_set:nn { enumext / keyans } { show-pos = #1 },
4080     show-pos .value_required:n = true,
4081     unknown .code:n = {
4082         \tl_set:Nn \l__enumext_envir_name_tl { keyanspic }
4083         \__enumext_keyans_unknown_keys:n {#1}
4084     },
4085 }

```

(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.

```

4086 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
4087 {
4088     \int_incr:N \l__enumext_keyans_pic_level_int
4089     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
4090     {
4091         \msg_error:nn { enumext } { keyanspic-nested }
4092     }
4093     \__enumext_keyans_name_and_start:
4094 }

```

Parse [`<key = val>`] for `keyanspic` environment.

```

4095 \cs_new_protected:Npn \__enumext_keyans_pic_parse_keys:n #1
4096 {
4097     \tl_if_novalue:nF {#1}
4098     {
4099         \keys_set:nn { enumext / keyanspic } {#1}
4100     }
4101 }

```

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

```

4102 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
4103 {
4104     \dim_compare:nNnT { #1 } < { \c_zero_dim }
4105     {
4106         \skip_set:Nn #1 { -#1 }
4107     }
4108 }

```


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 `\l__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.

```

4109 \cs_new_protected:Npn \__enumext_keyans_pic_arg_two:
4110 {
4111   \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
4112   \__enumext_list_arg_two_v:
4113   \__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 `\l__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*.

```

4114   \bool_if:NF \l__enumext_anspic_label_above_bool
4115   {
4116     \stepcounter { enumXv }
4117     \hbox_set:Nn \l__enumext_anspic_label_box { \l__enumext_label_v_tl }
4118     \dim_set:Nn \l__enumext_anspic_label_htdp_dim
4119     {
4120       \box_ht_plus_dp:N \l__enumext_anspic_label_box
4121     }
4122     \skip_add:Nn \parsep
4123     {
4124       \l__enumext_anspic_label_htdp_dim
4125       + \box_dp:N \strutbox
4126       + \l__enumext_anspic_label_sep_skip
4127     }
4128   }

```

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.

```

4129   \dim_add:Nn \leftmargin { -\l__enumext_labelwidth_v_dim - \l__enumext_labelsep_v_dim }
4130   \ignorespaces
4131   \skip_add:Nn \topsep { 0.5\box_dp:N \strutbox }
4132   \dim_zero:N \listparindent
4133   \skip_zero:N \partopsep
4134   \skip_zero:N \itemsep
4135 }

```

(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.

```

4136 \NewDocumentEnvironment{keyanspic}{ o }
4137 {
4138   \__enumext_keyans_pic_safe_exec:
4139   \__enumext_keyans_pic_parse_keys:n {#1}
4140   \begin{list} { } { \__enumext_keyans_pic_arg_two: }
4141   \IfDocumentMetadataT
4142   {
4143     \tag_suspend:n {list}
4144   }
4145   \item[] \scan_stop:
4146   \RenewDocumentCommand \item {}
4147   {
4148     \msg_error:nn { enumext } { keyanspic-item-cmd }
4149   }
4150   \IfDocumentMetadataT
4151   {
4152     \tag_resume:n {keyanspic}
4153     \tag_tool:n {para/tagging=false}
4154     \tag_suspend:n {keyanspic}
4155   }
4156 }

```

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.

```

4157 {
4158   \IfDocumentMetadataT
4159   {
4160     \tag_resume:n {keyanspic}
4161     \tag_mc_end_push:
4162     \tag_struct_begin:n {tag=L,attribute=enumerate}
4163   }
4164   \__enumext_anspic_exec:
4165   \IfDocumentMetadataT
4166   {
4167     \tag_suspend:n {keyanspic}
4168   }
4169   \end{list}
4170   \IfDocumentMetadataT
4171   {
4172     \tag_struct_end:n {tag=L}
4173     \tag_mc_begin_pop:n {}
4174     \tag_struct_end:n {tag=L}
4175     \tag_mc_begin_pop:n {}
4176   }

```

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

```

4177   \__enumext_check_starred_cmd:n { anspic }
4178   \setcounter { enumXvi } { 0 }
4179   \bool_if:NTF \__enumext_anspic_label_above_bool
4180   {
4181     \par\addvspace{ 0.5\box_dp:N \strutbox }
4182   }
4183   {
4184     \par
4185     \addvspace
4186     {
4187       \dim_eval:n
4188       {
4189         \__enumext_anspic_label_htdp_dim + \box_ht_plus_dp:N \strutbox
4190         + \__enumext_anspic_label_sep_skip + \__enumext_topsep_v_skip
4191       }
4192     }
4193   }
4194 }

```

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

13.42.2 The command `\anspic`

The `\anspic` command take three arguments, the *starred versions* `\anspic*[\langle content \rangle]` store the current `\label` next to the *optional argument* `[\langle content \rangle]` in the *sequence* and *prop list* defined by `save-ans` key. The third *mandatory argument* `{\langle drawing or tabular \rangle}` is NOT stored in the *sequence* or *prop list*.

- One of the complications here to make the `keyanspic` environment compatible with *tagged* PDF is the position of `\label`, the `\anspic` command processes the arguments in order, where #1 and #2 correspond to `\label` and #3 to the mandatory argument and puts all this inside a `minipage` environment. If #1 and #2, that is `\label`, is above #3 there are no problems with *tagged* PDF, but if #3 comes first the list created with *tagged* PDF will not be correct.

We check that the command is active in the `keyanspic` environment only if the `save-ans` key is present, otherwise we return an error. The three arguments are handled by the function `__enumext_anspic_args:nnn` and stored in the sequence `__enumext_anspic_args_seq` which is processed by the `keyanspic` environment.

```

4195 \NewDocumentCommand \anspic { s o +m }
4196 {
4197   \bool_if:NF \__enumext_store_active_bool
4198   {
4199     \msg_error:nnnn { enumext } { wrong-place } { keyanspic } { save-ans }
4200   }
4201   \int_compare:nNt { \__enumext_level_int } > { 1 }
4202   {
4203     \msg_error:nn { enumext } { keyanspic-wrong-level }
4204   }
4205   \int_compare:nNt { \__enumext_keyans_level_int } = { 1 }

```

```

4206     {
4207         \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
4208     }
4209     \seq_put_right:Nn \__enumext_anspic_args_seq
4210     {
4211         \__enumext_anspic_args:nnn { #1 } { #2 } { #3 }
4212     }
4213 }

```

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”.

```

4214 \cs_new_protected:Npn \__enumext_anspic_body_dim:n #1
4215 {
4216     \bool_if:NF \l__enumext_anspic_label_above_bool
4217     {
4218         \IfDocumentMetadataT
4219         {
4220             \tag_suspend:n {keyanspic}
4221         }
4222         \vbox_set:Nn \l__enumext_anspic_body_box { #1 }
4223         \dim_set:Nn \l__enumext_anspic_body_htdp_dim
4224         {
4225             \box_ht_plus_dp:N \l__enumext_anspic_body_box
4226         }
4227         \IfDocumentMetadataT
4228         {
4229             \tag_resume:n {keyanspic}
4230         }
4231     }
4232 }

```

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.

```

4233 \cs_new_protected:Npn \__enumext_anspic_label:nn #1 #2
4234 {
4235     \makebox[ \l__enumext_anspic_mini_width_dim ][ c ]
4236     {
4237         \bool_if:nTF { #1 }
4238         {
4239             \bool_set_true:N \l__enumext_item_wrap_key_bool
4240             \bool_set_true:N \l__enumext_wrap_label_v_bool
4241             \__enumext_keyans_save_item_opt:n { #2 }
4242             \__enumext_keyans_addto_prop:n { #2 }
4243             \__enumext_keyans_store_ref:
4244             \__enumext_keyans_addto_seq:n { #2 }
4245             \int_gincr:N \g__enumext_check_starred_cmd_int
4246             \__enumext_keyans_show_ans:
4247             \__enumext_keyans_show_pos:
4248             \makebox[ \l__enumext_labelwidth_v_dim ][ c ]
4249             {
4250                 \tl_use:N \l__enumext_label_font_style_v_tl
4251                 \__enumext_keyans_wrapper_label:n { \l__enumext_label_vi_tl }
4252             }
4253             \skip_horizontal:n { \l__enumext_labelsep_v_dim }
4254             \__enumext_keyans_show_item_opt:
4255         }
4256         {
4257             \bool_set_false:N \l__enumext_item_wrap_key_bool
4258             \tl_use:N \l__enumext_label_font_style_v_tl
4259             \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl }
4260         }
4261     }
4262 }

```

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.

```

4263 \cs_new_protected:Npn \__enumext_anspic_label_pos:nnn #1 #2 #3
4264 {
4265     \stepcounter { enumXvi }
4266     \__enumext_anspic_body_dim:n { #3 }
4267     \bool_if:NTF \l__enumext_anspic_label_above_bool

```

```

4268     {
4269       \__enumext_anspic_label:nn { #1 } { #2 }
4270     }
4271     {
4272       \raisebox
4273       {
4274         -\dim_eval:n
4275         {
4276           \__enumext_anspic_label_htdp_dim
4277           + \__enumext_anspic_body_htdp_dim
4278           + \box_dp:N \strutbox
4279           + \__enumext_anspic_label_sep_skip
4280         }
4281       }
4282       [ opt ] [ opt ]
4283       {
4284         \__enumext_anspic_label:nn { #1 } { #2 }
4285       }
4286     }
4287   }
4288   %

```

The `__enumext_anspic_args:nnn` function will be responsible for placing the code compatible with *tagged* PDF and the arguments within the `__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.

```

4289 \cs_new_protected:Nn \__enumext_anspic_args:nnn
4290 {
4291   \__enumext_anspic_start_list_tag:
4292   \__enumext_anspic_label_pos:nnn { #1 } { #2 } { #3 }
4293   \__enumext_anspic_stop_start_list_tag:
4294   \bool_if:NTF \__enumext_anspic_label_above_bool
4295   {
4296     \[\[ \__enumext_anspic_label_sep_skip] #3
4297   }
4298   {
4299     \[ #3
4300   }
4301   \__enumext_anspic_stop_list_tag:
4302 }

```

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`.

```

4303 \cs_new_protected:Nn \__enumext_anspic_print:n
4304 {
4305   \clist_map_function:nN { #1 } \__enumext_anspic_row:n
4306 }
4307 \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 `__enumext_anspic_args_seq` sequence inside them.

```

4308 \cs_new_protected:Nn \__enumext_anspic_row:n
4309 {
4310   \dim_set:Nn \__enumext_anspic_mini_width_dim { \linewidth / #1 }
4311   \int_set:Nn \__enumext_anspic_above_int { \__enumext_anspic_below_int }
4312   \int_set:Nn \__enumext_anspic_below_int { \__enumext_anspic_above_int + #1 }
4313   \int_step_inline:nnn
4314   { \__enumext_anspic_above_int + 1 }
4315   { \__enumext_anspic_below_int }
4316   {
4317     \IfDocumentMetadataT
4318     {
4319       \tag_suspend:n {minipage}
4320     }
4321     \begin{minipage}[ \__enumext_anspic_mini_pos_str ]{ \__enumext_anspic_mini_width_dim }
4322       \centering
4323       \seq_item:Nn \__enumext_anspic_args_seq { ##1 }
4324     \end{minipage}
4325     \IfDocumentMetadataT
4326     {
4327       \tag_resume:n {minipage}
4328     }
4329   }

```

```

4330 \par
4331 }

```

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*.

```

4332 \cs_new_protected:Nn \__enumext_anspic_exec:
4333 {
4334   \tl_if_empty:NTF \__enumext_anspic_layout_style_tl
4335   {
4336     \__enumext_anspic_print:e { \seq_count:N \__enumext_anspic_args_seq }
4337   }
4338   {
4339     \__enumext_anspic_print:V \__enumext_anspic_layout_style_tl
4340   }
4341 }

```

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

13.43 The horizontal environments

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

To achieve the *horizontal list environments* we will capture the `\item` command and the `\content` of this in *horizontal box* using `\makebox` for the `label` and a `minipage` environment for the `\content` passed to `\item`, we will also add the *optional argument* (`\number`) 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* (`\number`).

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 \TeX and any environment defined using base `list` environment, including: `itemize`, `enumerate`, `description`, `quote`, `quotation`, `verse`, `center`, `flushleft`, `flushright`, `verbatim`, `tabbing`, `trivlist`, `list` and all environments created with `\newtheorem`.

One way to get around this is to use something like:

```
\AddToHook{env/enumerate/before}{recover original \item definition}
```

inside `minipage`, but in my partial tests this does not have the desired effect and the vertical and horizontal spacing is distorted. For now this will remain as a limitation and I will see if it is feasible to implement it in the future.

For compatibility with the *tagged* PDF we close the environments according to the presence or not of the `mini-env` key.

13.43.1 Functions for item box width

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

```

\__enumext_starred_columns_set_vii:
\__enumext_starred_columns_set_viii:

```

```

4342 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
4343 {
4344   \dim_compare:nNnT { \__enumext_columns_sep_vii_dim } = { \c_zero_dim }
4345   {
4346     \dim_set:Nn \__enumext_columns_sep_vii_dim
4347     {
4348       ( \__enumext_labelwidth_vii_dim + \__enumext_labelsep_vii_dim )
4349       / \__enumext_columns_vii_int
4350     }
4351   }
4352   \int_set:Nn \__enumext_tmpa_vii_int { \__enumext_columns_vii_int - 1 }
4353   \dim_set:Nn \__enumext_item_width_vii_dim
4354   {
4355     ( \linewidth - \__enumext_columns_sep_vii_dim * \__enumext_tmpa_vii_int )
4356     / \__enumext_columns_vii_int
4357     - \__enumext_labelwidth_vii_dim
4358     - \__enumext_labelsep_vii_dim
4359   }

```

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

```

4360   \dim_compare:nNnT { \__enumext_rightmargin_vii_dim } > { \c_zero_dim }
4361   {
4362     \dim_sub:Nn \__enumext_item_width_vii_dim
4363     {
4364       ( \__enumext_rightmargin_vii_dim * \__enumext_tmpa_vii_int )
4365       / \__enumext_columns_vii_int

```

```

4366     }
4367     \dim_add:Nn \l__enumext_columns_sep_vii_dim
4368     {
4369         \l__enumext_rightmargin_vii_dim
4370     }
4371 }
4372 }

```

Same implementation for the `keyans*` environment.

```

4373 \cs_new_protected:Nn \l__enumext_starred_columns_set_viii:
4374 {
4375     \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
4376     {
4377         \dim_set:Nn \l__enumext_columns_sep_viii_dim
4378         {
4379             ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
4380             / \l__enumext_columns_viii_int
4381         }
4382     }
4383     \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - 1 }
4384     \dim_set:Nn \l__enumext_item_width_viii_dim
4385     {
4386         ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
4387         / \l__enumext_columns_viii_int
4388         - \l__enumext_labelwidth_viii_dim
4389         - \l__enumext_labelsep_viii_dim
4390     }
4391     \dim_compare:nNnT { \l__enumext_rightmargin_viii_dim } > { \c_zero_dim }
4392     {
4393         \dim_sub:Nn \l__enumext_item_width_viii_dim
4394         {
4395             ( \l__enumext_rightmargin_viii_dim * \l__enumext_tmpa_vii_int )
4396             / \l__enumext_columns_viii_int
4397         }
4398         \dim_add:Nn \l__enumext_columns_sep_viii_dim
4399         {
4400             \l__enumext_rightmargin_viii_dim
4401         }
4402     }
4403 }

```

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

13.43.2 Functions for join item columns

```

\__enumext_starred_joined_item_vii:n
\__enumext_starred_joined_item_viii:n

```

The functions `__enumext_starred_joined_item_vii:n` and `__enumext_starred_joined_item_viii:n` will set the *width* of the box in which the `⟨content⟩` passed to `\item(⟨columns⟩)` will be stored together with the value of `\itemwidth` for the `enumext*` environment.

```

4404 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
4405 {
4406     \int_set:Nn \l__enumext_joined_item_vii_int {#1}
4407     \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
4408     {
4409         \msg_warning:nnee { enumext } { item-joined }
4410         { \int_use:N \l__enumext_joined_item_vii_int }
4411         { \int_use:N \l__enumext_columns_vii_int }
4412         \int_set:Nn \l__enumext_joined_item_vii_int
4413         {
4414             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4415         }
4416     }
4417     \int_compare:nNnT
4418     { \l__enumext_joined_item_vii_int }
4419     >
4420     { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4421     {
4422         \msg_warning:nnee { enumext } { item-joined-columns }
4423         { \int_use:N \l__enumext_joined_item_vii_int }
4424         {
4425             \int_eval:n
4426             { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4427         }

```

```

4428         \int_set:Nn \l__enumext_joined_item_vii_int
4429         {
4430             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4431         }
4432     }
4433     \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { 1 }
4434     {
4435         \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
4436         \int_decr:N \l__enumext_joined_item_aux_vii_int
4437         \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
4438         \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
4439         \dim_set:Nn \l__enumext_joined_width_vii_dim
4440         {
4441             \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
4442             + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
4443                 + \l__enumext_columns_sep_vii_dim
4444                 )*\l__enumext_joined_item_aux_vii_int
4445         }
4446         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
4447     }
4448     {
4449         \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
4450         \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
4451     }
4452 }

```

Same implementation for the `keyans*` environment.

```

4453 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
4454 {
4455     \int_set:Nn \l__enumext_joined_item_viii_int {#1}
4456     \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
4457     {
4458         \msg_warning:nnee { enumext } { item-joined }
4459         { \int_use:N \l__enumext_joined_item_viii_int }
4460         { \int_use:N \l__enumext_columns_viii_int }
4461         \int_set:Nn \l__enumext_joined_item_viii_int
4462         {
4463             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4464         }
4465     }
4466     \int_compare:nNnT
4467     { \l__enumext_joined_item_viii_int }
4468     >
4469     { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4470     {
4471         \msg_warning:nnee { enumext } { item-joined-columns }
4472         { \int_use:N \l__enumext_joined_item_viii_int }
4473         {
4474             \int_eval:n
4475             { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4476         }
4477         \int_set:Nn \l__enumext_joined_item_viii_int
4478         {
4479             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4480         }
4481     }
4482     \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { 1 }
4483     {
4484         \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
4485         \int_decr:N \l__enumext_joined_item_aux_viii_int
4486         \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
4487         \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
4488         \dim_set:Nn \l__enumext_joined_width_viii_dim
4489         {
4490             \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
4491             + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
4492                 + \l__enumext_columns_sep_viii_dim
4493                 )*\l__enumext_joined_item_aux_viii_int
4494         }
4495         \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
4496     }
4497     {

```



```

4498         \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
4499         \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
4500     }
4501 }

```

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

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

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`.

```

4502 \cs_new_protected:Nn \__enumext_start_mini_vii:
4503 {
4504     \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
4505     {
4506         \dim_set:Nn \l__enumext_minipage_left_vii_dim
4507         {
4508             \linewidth
4509             - \l__enumext_minipage_right_vii_dim
4510             - \l__enumext_minipage_hsep_vii_dim
4511         }
4512         \bool_set_true:N \l__enumext_minipage_active_vii_bool
4513         \dim_gset_eq:NN
4514             \g__enumext_minipage_right_vii_dim
4515             \l__enumext_minipage_right_vii_dim
4516         \__enumext_mini_addvspace_vii:
4517         \nointerlineskip\noindent
4518         \__enumext_mini_page{ \l__enumext_minipage_left_vii_dim }
4519     }
4520 }

```

The function `__enumext_stop_mini_vii:` closes the `__enumext_mini_page` environment on the “left side”, applies `\hfill` and set the variable `\g__enumext_minipage_active_vii_bool` to “true” which will be used in the function `__enumext_after_env:nn` to execute the `minipage` on the “right side”. At this point we will execute the `__enumext_stop_list:` and `__enumext_stop_store_level_vii:` functions stopping the `list` environment and the level saving mechanism for storage in *sequence* of the `\anskey` command and `anskey*` environment. This function is passed to the `__enumext_after_list_vii:` function in the second part of the `enumext*` environment definition (§13.44).

```

4521 \cs_new_protected:Nn \__enumext_stop_mini_vii:
4522 {
4523     \bool_if:NTF \l__enumext_minipage_active_vii_bool
4524     {
4525         \__enumext_stop_list:
4526         \__enumext_stop_store_level_vii:
4527         \IfDocumentMetadataT { \tag_resume:n {enumext*} }
4528         \end__enumext_mini_page
4529         \hfill
4530         \bool_gset_true:N \g__enumext_minipage_active_vii_bool
4531     }
4532     {
4533         \__enumext_stop_list:
4534         \__enumext_stop_store_level_vii:
4535     }
4536 }

```

(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.

```

4537 \__enumext_after_env:nn {enumext*}
4538 {
4539     \bool_if:NT \g__enumext_minipage_active_vii_bool
4540     {
4541         \__enumext_minipage:w [ t ] { \g__enumext_minipage_right_vii_dim }
4542         \legacy_if_gset_false:n { @minipage }
4543         \skip_vertical:N \c_zero_skip
4544         \par\addvspace { \g__enumext_minipage_right_skip }
4545         \bool_if:NF \g__enumext_minipage_center_vii_bool

```

```

4546         {
4547             \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
4548             {
4549                 \centering
4550             }
4551         }
4552     \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
4553     {
4554         \tl_use:N \g__enumext_miniright_code_vii_tl
4555     }
4556     \box_use_drop:N \l__enumext_miniright_code_vii_box
4557     \skip_vertical:N \c_zero_skip
4558     \__enumext_endminipage:
4559     \par\addvspace{ \g__enumext_minipage_after_skip }
4560 }
4561 \bool_gset_false:N \g__enumext_minipage_active_vii_bool
4562 \bool_gset_true:N \g__enumext_minipage_center_vii_bool
4563 \tl_gclear:N \g__enumext_miniright_code_vii_tl
4564 \dim_gzero:N \g__enumext_minipage_right_vii_dim
4565 \bool_gset_false:N \g__enumext_starred_bool
4566 }

```

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

`__enumext_stop_mini_viii:`

```

4567 \cs_new_protected:Nn \__enumext_start_mini_viii:
4568 {
4569     \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
4570     {
4571         \dim_set:Nn \l__enumext_minipage_left_viii_dim
4572         {
4573             \linewidth
4574             - \l__enumext_minipage_right_viii_dim
4575             - \l__enumext_minipage_hsep_viii_dim
4576         }
4577         \bool_set_true:N \l__enumext_minipage_active_viii_bool
4578         \dim_gset_eq:NN
4579             \g__enumext_minipage_right_viii_dim
4580             \l__enumext_minipage_right_viii_dim
4581         \__enumext_mini_addvspace_viii:
4582         \nointerlineskip\noindent
4583         \__enumext_mini_page{ \l__enumext_minipage_left_viii_dim }
4584     }
4585 }
4586 \cs_new_protected:Nn \__enumext_stop_mini_viii:
4587 {
4588     \bool_if:NTF \l__enumext_minipage_active_viii_bool
4589     {
4590         \__enumext_stop_list:
4591         \IfDocumentMetadataTF { \tag_resume:n {keyans*} } { }
4592         \end__enumext_mini_page
4593         \hfill
4594         \bool_gset_true:N \g__enumext_minipage_active_viii_bool
4595     }
4596     {
4597         \__enumext_stop_list:
4598     }
4599 }
4600 \__enumext_after_env:nn {keyans*}
4601 {
4602     \bool_if:NT \g__enumext_minipage_active_viii_bool
4603     {
4604         \__enumext_mini_page{ \g__enumext_minipage_right_viii_dim }
4605         \par\addvspace { \g__enumext_minipage_right_skip }
4606         \bool_if:NF \g__enumext_minipage_center_viii_bool
4607         {
4608             \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
4609             {
4610                 \centering
4611             }
4612         }
4613         \vbox_set_top:Nn \l__enumext_miniright_code_viii_box

```

```

4614         {
4615             \tl_use:N \g__enumext_miniright_code_viii_tl
4616         }
4617         \box_use_drop:N \l__enumext_miniright_code_viii_box
4618         \end__enumext_mini_page
4619         \par\addvspace{ \g__enumext_minipage_after_skip }
4620     }
4621     \bool_gset_false:N \g__enumext_minipage_active_viii_bool
4622     \bool_gset_true:N \g__enumext_minipage_center_viii_bool
4623     \tl_gclear:N \g__enumext_miniright_code_viii_tl
4624     \dim_gzero:N \g__enumext_minipage_right_viii_dim
4625 }

```

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

13.44 The environment enumext*

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

```

4626 \NewDocumentEnvironment{enumext*}{o }
4627 {
4628     \__enumext_safe_exec_vii:
4629     \__enumext_parse_keys_vii:n {#1}
4630     \__enumext_before_list_vii:
4631     \__enumext_start_store_level_vii:
4632     \__enumext_start_list:nn { }
4633     {
4634         \__enumext_list_arg_two_vii:
4635         \__enumext_before_keys_exec_vii:
4636     }
4637     \setcounter { enumXvii } { \int_eval:n { \int_use:c { l__enumext_start_vii_int } - 1 } }
4638     \IfDocumentMetadataT { \tag_suspend:n {enumext*} }
4639     \__enumext_starred_columns_set_vii:
4640     \item[] \scan_stop:
4641     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_first_item_tmp_vii:
4642     \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
4643     \ignorespaces
4644 }
4645 {
4646     \IfDocumentMetadataT { \tag_struct_end:n {tag=text-unit} }
4647     \__enumext_stop_item_tmp_vii:
4648     \__enumext_remove_extra_parsep_vii:
4649     \__enumext_after_list_vii:
4650 }

```

(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.

```

4651 \cs_new_protected:Nn \__enumext_safe_exec_vii:
4652 {
4653     \__enumext_is_not_nested:
4654     \__enumext_internal_mini_page:
4655     \int_incr:N \l__enumext_level_h_int
4656     \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
4657     {
4658         \msg_error:nn { enumext } { nested }
4659     }
4660     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
4661     {
4662         \msg_error:nnn { enumext } { nested-horizontal } { keyans* }
4663     }
4664     \bool_set_true:N \l__enumext_starred_bool

```

```

4665     \bool_set_false:N \l__enumext_standar_bool
4666     \__enumext_is_on_first_level:
4667 }

```

(End of definition for __enumext_safe_exec_vii:.)

__enumext_parse_keys_vii:n

First we will clear the variable \l__enumext_series_str used by the key `series`, process the environment [`<key = val>`] and execute the function __enumext_parse_series:n and used by the key `series`, then we execute the function __enumext_store_active_keys_vii:n and reprocess the (`keys`) to pass them to the storage *sequence* if the key `save-key` is not active.

```

4668 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
4669 {
4670     \tl_if_novalue:nF {#1}
4671     {
4672         \str_clear:N \l__enumext_series_str
4673         \keys_set:nn { enumext / enumext* } {#1}
4674         \__enumext_parse_series:n {#1}
4675         \__enumext_store_active_keys_vii:n {#1}
4676     }
4677 }

```

(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*`.

```

4678 \cs_new_protected:Nn \__enumext_before_list_vii:
4679 {
4680     \__enumext_vspace_above_vii:
4681     \__enumext_check_ans_active:
4682     \__enumext_before_args_exec_vii:
4683     \__enumext_start_mini_vii:
4684 }

```

(End of definition for __enumext_before_list_vii:.)

__enumext_after_list_vii:

The function __enumext_after_list_vii: first calls the function __enumext_stop_mini_vii: which internally calls __enumext_stop_list: and __enumext_stop_store_level_vii: (§13.43.3) used by the keys `mini-env`, `mini-right` and `mini-right*`, then to the functions __enumext_after_stop_list_vii: used by the key `after`, __enumext_check_ans_key_hook: used by the key `check-ans`, __enumext_vspace_below_vii: used by the keys `below` and `below*`. Finally set \l__enumext_starred_bool to false and call the __enumext_resume_save_counter: function used by the `series`, `resume` and `resume*` keys.

```

4685 \cs_new_protected:Nn \__enumext_after_list_vii:
4686 {
4687     \__enumext_stop_mini_vii:
4688     \__enumext_after_stop_list_vii:
4689     \__enumext_check_ans_key_hook:
4690     \__enumext_vspace_below_vii:
4691     \bool_set_false:N \l__enumext_starred_bool
4692     \__enumext_resume_save_counter:
4693 }

```

(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.

```

4694 \cs_new_protected:Nn \__enumext_start_store_level_vii:
4695 {
4696     \bool_if:NT \l__enumext_store_active_bool
4697     {
4698         \int_compare:nNnT { \l__enumext_level_int } > { 0 }
4699         {
4700             \__enumext_store_level_open_vii:
4701         }
4702     }
4703 }
4704 \cs_new_protected:Nn \__enumext_stop_store_level_vii:

```

```

4705 {
4706   \bool_if:NT \l__enumext_store_active_bool
4707   {
4708     \int_compare:nNnT { \l__enumext_level_int } > { 0 }
4709     {
4710       \__enumext_store_level_close_vii:
4711     }
4712   }
4713 }

```

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

13.44.1 The command \item in enumext*

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

```

4714 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_vii:
4715 {
4716   \skip_horizontal:n
4717   {
4718     -\l__enumext_labelwidth_vii_dim - \l__enumext_labelsep_vii_dim
4719   }
4720   \ignorespaces
4721 }

```

(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
\__enumext_starred_item_vii_aux_i:w
\__enumext_starred_item_vii_aux_ii:w
\__enumext_starred_item_vii_aux_iii:w

```

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

```

4722 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
4723 {
4724   \__enumext_stop_item_tmp_vii:
4725   \int_incr:N \l__enumext_item_column_pos_vii_int
4726   \int_gincr:N \g__enumext_item_count_all_vii_int
4727   \__enumext_item_peek_args_vii:
4728 }

```

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).

```

4729 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
4730 {
4731   \peek_meaning:NTF (
4732     { \__enumext_joined_item_vii:w }
4733     { \__enumext_joined_item_vii:w (1) }
4734   }

```

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.

```

4735 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
4736 {
4737   \__enumext_starred_joined_item_vii:n {#1}
4738   \peek_meaning_remove:NTF *
4739   { \__enumext_starred_item_vii:w }
4740   { \__enumext_standar_item_vii:w }
4741 }

```

The function __enumext_standar_item_vii:w will first look for the argument “[”, if present it will set the state of the variable \l__enumext_wrap_label_opt_vii_bool equal to the state of the variable \l__enumext_wrap_label_opt_vii_bool handled by the key wrap-label* and finally execute the non-enumerated version \item[<custom>] by means of the function __enumext_start_item_vii:w, otherwise we will set the value of the variable \l__enumext_wrap_label_vii_bool handled by the wrap-label key to true and set the switch \if@noitemarg to true to execute the enumerated version of \item by means of the function __enumext_start_item_vii:w [\l__enumext_label_vii_tl].

```

4742 \cs_new_protected:Npn \__enumext_standar_item_vii:w
4743 {

```

```

4744 \bool_set_false:N \l__enumext_item_starred_vii_bool
4745 \peek_meaning:NTF [
4746 {
4747   \bool_set_eq:NN \l__enumext_wrap_label_vii_bool \l__enumext_wrap_label_opt_vii_bool
4748   \l__enumext_start_item_vii:w
4749 }
4750 {
4751   \bool_set_true:N \l__enumext_wrap_label_vii_bool
4752   \legacy_if_set_true:n { @noitemarg }
4753   \l__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ] \ignorespaces
4754 }
4755 }

```

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

```

4756 \cs_new_protected:Npn \l__enumext_starred_item_vii:w
4757 {
4758   \bool_set_true:N \l__enumext_item_starred_vii_bool
4759   \bool_set_true:N \l__enumext_wrap_label_vii_bool
4760   \peek_meaning:NTF [
4761     { \l__enumext_starred_item_vii_aux_i:w }
4762     { \l__enumext_starred_item_vii_aux_ii:w }
4763   ]
4764   \cs_new_protected:Npn \l__enumext_starred_item_vii_aux_i:w [#1]
4765   {
4766     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
4767     \l__enumext_starred_item_vii_aux_ii:w
4768   }
4769   \cs_new_protected:Npn \l__enumext_starred_item_vii_aux_ii:w
4770   {
4771     \peek_meaning:NTF [
4772       { \l__enumext_starred_item_vii_aux_iii:w }
4773       {
4774         \dim_set_eq:NN \l__enumext_item_symbol_sep_vii_dim \l__enumext_labelsep_vii_dim
4775         \legacy_if_set_true:n { @noitemarg }
4776         \l__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ] \ignorespaces
4777       }
4778     ]
4779     \cs_new_protected:Npn \l__enumext_starred_item_vii_aux_iii:w [#1]
4780     {
4781       \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
4782       \legacy_if_set_true:n { @noitemarg }
4783       \l__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ] \ignorespaces
4784     }

```

(End of definition for `\l__enumext_start_item_tmp_vii:` and others.)

`\l__enumext_fake_make_label_vii:n`

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

- For compatibility with *tagged* PDF and `hyperref` when an environment `enumext` is nested in `enumext*` and the key `save-ans` is not active need setting the `\if@hyper@item` switch to “true”. The explanation for this is given by the master Heiko Oberdiek on `\refstepcounter{enumi}` twice (or more) creates destination with the same identifier. This patch is only needed if you are running `pdflatex` and not if you are running `lua1latex`

```

4785 \cs_new_protected_nopar:Npn \l__enumext_fake_make_label_vii:n #1
4786 {
4787   \legacy_if:nT { @noitemarg }
4788   {
4789     \legacy_if_set_false:n { @noitemarg }
4790     \legacy_if:nT { @nmbrrlist }
4791     {
4792       \IfDocumentMetadataT
4793       {
4794         \bool_if:NT \l__enumext_hyperref_bool
4795         {
4796           \legacy_if_set_true:n { @hyper@item }

```

```

4797         }
4798     }
4799     \refstepcounter{enumXvii}
4800     \bool_if:NT \l__enumext_check_answers_bool
4801     {
4802         \int_gincr:N \g__enumext_item_number_int
4803         \bool_set_true:N \l__enumext_item_number_bool
4804     }
4805 }
4806 }
4807 \bool_if:NT \l__enumext_item_starred_vii_bool
4808 {
4809     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
4810     {
4811         \tl_gset_eq:NN
4812         \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
4813     }
4814     \mode_leave_vertical:
4815     \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
4816     \hbox_overlap_left:n { \g__enumext_item_symbol_aux_vii_tl }
4817     \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
4818     \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
4819 }
4820 \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
4821 {
4822     \tl_use:N \l__enumext_label_font_style_vii_tl
4823     \bool_if:NTF \l__enumext_wrap_label_vii_bool
4824     {
4825         \__enumext_wrapper_label_vii:n {#1}
4826     }
4827     { #1 }
4828 }
4829 \skip_horizontal:N \l__enumext_labelsep_vii_dim \ignorespaces
4830 }

```

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

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

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

```

\__enumext_start_item_vii:w
\__enumext_stop_item_vii:

```

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`.

```

4831 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
4832 {
4833     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
4834     \hbox_set_to_wd:Nnw \l__enumext_item_text_vii_box
4835     {
4836         \l__enumext_joined_width_vii_dim
4837         + \l__enumext_labelwidth_vii_dim
4838         + \l__enumext_labelsep_vii_dim
4839     }

```

Redefine the `\footnote` command.

```
4840 \__enumext_renew_footnote_starred:
```

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

```

4841 \__enumext_start_list_tag:n {enumext*}
4842 \__enumext_fake_make_label_vii:n {#1}
4843 \__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.

```

4844 \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
4845 \dim_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
4846 \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip

```



```

4847     \__enumext_unskip_unkern:
4848     \__enumext_unskip_unkern:
4849     \skip_horizontal:n { -\l__enumext_listparindent_vii_dim } \ignorespaces
4850     \tl_use:N \l__enumext_fake_item_indent_vii_tl
4851     \tl_use:N \l__enumext_after_list_args_vii_tl
4852 }

```

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*.

```

4853 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
4854 {
4855     \__enumext_endminipage:
4856     \__enumext_stop_list_tag:n {enumext*}
4857     \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*.

```

4858     \int_set:Nn \hbadness { 10000 }
4859     \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`.

```

4860     \int_compare:nNnTF
4861     { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
4862     {
4863         \par\noindent
4864         \int_zero:N \l__enumext_item_column_pos_vii_int
4865     }
4866     {
4867         \skip_horizontal:N \l__enumext_columns_sep_vii_dim
4868     }
4869 }

```

(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*.

```

4870 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
4871 {
4872     \int_compare:nNnT
4873     {
4874         \int_mod:nn
4875         { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
4876     }
4877     =
4878     { 0 }
4879     {
4880         \para_end:
4881         \skip_vertical:n { -\l__enumext_itemsep_vii_skip }
4882         \skip_vertical:N \c_zero_skip
4883         \int_gzero:N \g__enumext_item_count_all_vii_int
4884     }
4885 }

```

(End of definition for `__enumext_remove_extra_parsep_vii:.`)

As we don’t want our check to be executed `check-ans` by levels but on the complete list, we will take it out of the `enumext*` environment using the “*hook*” function `__enumext_after_env:nn`.

```

4886 \__enumext_after_env:nn {enumext*}
4887 {
4888     \__enumext_execute_after_env:
4889 }

```

13.45 The environment keyans*

keyans* The implementation of **keyans*** environment is the similar as that used by the **enumext*** environment except for the `__enumext_check_starred_cmd:n` function added in the second part.

```

4890 \NewDocumentEnvironment{keyans*}{ o }
4891 {
4892   \__enumext_safe_exec_viii:
4893   \__enumext_parse_keys_viii:n {#1}
4894   \__enumext_before_list_viii:
4895   \__enumext_start_list:nn { }
4896   {
4897     \__enumext_list_arg_two_viii:
4898     \__enumext_before_keys_exec_viii:
4899   }
4900   \setcounter { enumXviii } { \int_eval:n { \int_use:c { \__enumext_start_viii_int } - 1 } }
4901   \IfDocumentMetadataT { \tag_suspend:n {keyans*} }
4902   \__enumext_starred_columns_set_viii:
4903   \item[] \scan_stop:
4904   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_first_item_tmp_viii:
4905   \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
4906   \ignorespaces
4907 }
4908 {
4909   \IfDocumentMetadataT { \tag_struct_end:n {tag=text-unit} }
4910   \__enumext_stop_item_tmp_viii:
4911   \__enumext_remove_extra_parsep_viii:
4912   \__enumext_check_starred_cmd:n { item }
4913   \__enumext_after_list_viii:
4914 }

```

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

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

```

4915 \cs_new_protected:Nn \__enumext_safe_exec_viii:
4916 {
4917   \bool_if:NF \l__enumext_store_active_bool
4918   {
4919     \msg_error:nnnn { enumext } { wrong-place } { keyans* } { save-ans }
4920   }
4921   \int_incr:N \l__enumext_keyans_level_h_int
4922   \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
4923   {
4924     \msg_error:nn { enumext } { nested }
4925   }
4926   \__enumext_keyans_name_and_start:
4927   \bool_if:NT \l__enumext_starred_bool
4928   {
4929     \msg_error:nnn { enumext } { nested-horizontal } { enumext* }
4930   }
4931   \bool_set_true:N \l__enumext_starred_bool
4932   % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
4933   \bool_set_false:N \l__enumext_store_active_bool
4934   \int_compare:nNnT { \l__enumext_level_int } > { 1 }
4935   {
4936     \msg_error:nn { enumext } { keyans-wrong-level }
4937   }
4938 }

```

(End of definition for `__enumext_safe_exec_viii:`.)

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

```

4939 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
4940 {
4941   \tl_if_novalue:nF {#1}
4942   {

```

```

4943     \keys_set:nn { enumext / keyans* } {#1}
4944   }
4945 }

```

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

`__enumext_before_list_viii:` The function `__enumext_before_list_viii:` will add the vertical spacing on the environment if the `above` key is active next to the `{\code}` defined by the `before*` key if it is active, the call the function `__enumext_start_mini_viii:` handle by `mini-env`.

```

4946 \cs_new_protected:Nn \__enumext_before_list_viii:
4947 {
4948   \__enumext_vspace_above_viii:
4949   \__enumext_before_args_exec_viii:
4950   \__enumext_start_mini_viii:
4951 }

```

(End of definition for `__enumext_before_list_viii:.`)

`__enumext_after_list_viii:` The function `__enumext_after_list_viii:` first call the function `__enumext_stop_mini_viii:`, then apply the `{\code}` handled by the `after` key together with the *vertical space* handled by the `below` key if they are present.

```

4952 \cs_new_protected:Nn \__enumext_after_list_viii:
4953 {
4954   \__enumext_stop_mini_viii:
4955   \__enumext_after_stop_list_viii:
4956   \__enumext_vspace_below_viii:
4957 }

```

(End of definition for `__enumext_after_list_viii:.`)

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

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

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

```

4958 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_viii:
4959 {
4960   \skip_horizontal:n
4961   {
4962     -\__enumext_labelwidth_viii_dim - \__enumext_labelsep_viii_dim
4963   }
4964   \ignorespaces
4965 }

```

(End of definition for `__enumext_first_item_tmp_viii:.`)

`__enumext_start_item_tmp_viii:` First we will call the function `__enumext_stop_item_tmp_viii:` that we will redefine later, we will increment the value of `\l__enumext_item_column_pos_viii_int` that will count the item’s by rows and the value of `\g__enumext_item_count_all_viii_int` that will count the total of item’s in the environment. `__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`.

```

4966 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
4967 {
4968   \__enumext_stop_item_tmp_viii:
4969   \int_incr:N \l__enumext_item_column_pos_viii_int
4970   \int_gincr:N \g__enumext_item_count_all_viii_int
4971   \__enumext_item_peek_args_viii:
4972 }

```

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).

```

4973 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
4974 {
4975   \peek_meaning:NTF (

```

```

4976     { \__enumext_joined_item_viii:w }
4977     { \__enumext_joined_item_viii:w (1) }
4978 }

```

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`.

```

4979 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
4980 {
4981   \__enumext_starred_joined_item_viii:n {#1}
4982   \peek_meaning_remove:NTF *
4983   { \__enumext_starred_item_viii:w }
4984   { \__enumext_standar_item_viii:w }
4985 }

```

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]`.

```

4986 \cs_new_protected:Npn \__enumext_standar_item_viii:w
4987 {
4988   \bool_set_false:N \l__enumext_item_starred_viii_bool
4989   \bool_set_false:N \l__enumext_item_wrap_key_bool
4990   \peek_meaning:NTF [
4991   {
4992     \bool_set_eq:NN \l__enumext_wrap_label_viii_bool \l__enumext_wrap_label_opt_viii_bool
4993     \__enumext_start_item_viii:w
4994   }
4995   {
4996     \bool_set_true:N \l__enumext_wrap_label_viii_bool
4997     \legacy_if_set_true:n { @noitemarg }
4998     \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ] \ignorespaces
4999   }
5000 }

```

(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⟩]`.

```

5001 \cs_new_protected:Npn \__enumext_starred_item_viii:w
5002 {
5003   \bool_set_true:N \l__enumext_item_starred_viii_bool
5004   \bool_set_true:N \l__enumext_item_wrap_key_bool
5005   \bool_set_true:N \l__enumext_wrap_label_viii_bool
5006   \peek_meaning:NTF [
5007   { \__enumext_starred_item_viii_aux_i:w }
5008   { \__enumext_starred_item_viii_aux_ii:w }
5009 }

```

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`.

```

5010 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
5011 {
5012   \tl_clear:N \l__enumext_store_current_label_tl
5013   \tl_if_novalue:nF { #1 }
5014   {
5015     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_viii_tl
5016     {
5017       \tl_put_right:NV \l__enumext_store_current_label_tl \l__enumext_store_keyans_item_opt
5018       \tl_put_right:Nn \l__enumext_store_current_label_tl { #1 }
5019     }
5020     \tl_set:Nn \l__enumext_store_current_opt_arg_tl { #1 }
5021   }
5022   \__enumext_starred_item_viii_aux_ii:w
5023 }

```

```

5024 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
5025 {
5026   \legacy_if_set_true:n { @noitemarg }
5027   \__enumext_start_item_viii:w [ \__enumext_label_viii_tl ] \ignorespaces
5028 }

```

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.

```

5029 \cs_new_protected:Nn \__enumext_keyans_starred_item_star:
5030 {
5031   \tl_put_left:Ne \l__enumext_store_current_label_tl { \l__enumext_label_viii_tl }
5032   \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
5033   \__enumext_keyans_store_ref:
5034   \tl_put_left:Nn \l__enumext_store_current_label_tl { \item }
5035   \__enumext_keyans_addto_seq_link:
5036   \int_gincr:N \g__enumext_check_starred_cmd_int
5037   \dim_compare:nNnT { \l__enumext_mark_sym_sep_viii_dim } = { \c_zero_dim }
5038   {
5039     \dim_set:Nn \l__enumext_mark_sym_sep_viii_dim { \l__enumext_labelsep_viii_dim }
5040   }
5041   \bool_if:NT \l__enumext_show_answer_bool
5042   {
5043     \tl_set_eq:NN \l__enumext_mark_answer_sym_tl \l__enumext_mark_answer_sym_viii_tl
5044     \str_set_eq:NN \l__enumext_mark_position_str \l__enumext_mark_position_viii_str
5045     \__enumext_print_keyans_box:NN
5046     \l__enumext_labelwidth_viii_dim \l__enumext_mark_sym_sep_viii_dim
5047   }
5048   \bool_if:NT \l__enumext_show_position_bool
5049   {
5050     \tl_set:Ne \l__enumext_mark_answer_sym_tl
5051     {
5052       \group_begin:
5053       \exp_not:N \normalfont
5054       \exp_not:N \footnotesize [ \int_eval:n
5055       {
5056         \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
5057       }
5058       ]
5059       \group_end:
5060     }
5061     \str_set_eq:NN \l__enumext_mark_position_str \l__enumext_mark_position_viii_str
5062     \__enumext_print_keyans_box:NN
5063     \l__enumext_labelwidth_viii_dim \l__enumext_mark_sym_sep_viii_dim
5064   }
5065 }

```

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

```

\__enumext_keyans_wrapper_label_viii:n
\__enumext_fake_make_label_viii:n

```

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

```

5066 \cs_new_protected:Npn \__enumext_keyans_wrapper_label_viii:n #1
5067 {
5068   \bool_lazy_all:nT
5069   {
5070     { \bool_if_p:N \l__enumext_wrap_label_viii_bool }
5071     { \bool_if_p:N \l__enumext_show_answer_bool }
5072     { \bool_if_p:N \l__enumext_item_wrap_key_bool }
5073     { \cs_if_exist_p:N \__enumext_keyans_wrapper_item_viii:n }
5074   }
5075   {
5076     \cs_set_eq:NN
5077     \__enumext_wrapper_label_viii:n \__enumext_keyans_wrapper_item_viii:n
5078   }
5079   \bool_if:NTF \l__enumext_wrap_label_viii_bool
5080   {
5081     \__enumext_wrapper_label_viii:n {#1}
5082   }
5083   { #1 }
5084 }
5085 \cs_new_protected_nopar:Npn \__enumext_fake_make_label_viii:n #1

```

```

5086 {
5087   \legacy_if:nT { @noitemarg }
5088   {
5089     \legacy_if_set_false:n { @noitemarg }
5090     \legacy_if:nT { @nmbrrlist }
5091     {
5092       \refstepcounter{enumXviii}
5093     }
5094   }
5095   \bool_if:NT \__enumext_item_starred_viii_bool
5096   {
5097     \__enumext_keyans_starred_item_star:
5098   }
5099   \makebox[ \__enumext_labelwidth_viii_dim ][ \__enumext_align_label_viii_str ]
5100   {
5101     \tl_use:N \__enumext_label_font_style_viii_tl
5102     \__enumext_keyans_wrapper_label_viii:n {#1}
5103   }
5104   \skip_horizontal:N \__enumext_labelsep_viii_dim \ignorespaces
5105 }

```

(End of definition for __enumext_keyans_wrapper_label_viii:n and __enumext_fake_make_label_viii:n)

13.45.2 Real definition of \item in keyans*

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

```

\__enumext_start_item_viii:w
\__enumext_stop_item_viii:
5106 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
5107 {
5108   \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
5109   \hbox_set_to_wd:Nnw \__enumext_item_text_viii_box
5110   {
5111     \__enumext_joined_width_viii_dim
5112     + \__enumext_labelwidth_viii_dim
5113     + \__enumext_labelsep_viii_dim
5114   }
5115   \__enumext_renew_footnote_starred:
5116   \__enumext_start_list_tag:n {keyans*}
5117   \__enumext_fake_make_label_viii:n {#1}
5118   \__enumext_stop_start_list_tag:
5119   \__enumext_minipage:w [ t ]{ \__enumext_joined_width_viii_dim }
5120   \dim_set_eq:NN \parindent \__enumext_listparindent_viii_dim
5121   \skip_set_eq:NN \parskip \__enumext_parsep_viii_skip
5122   \__enumext_unskip_unkern:
5123   \__enumext_unskip_unkern:
5124   \skip_horizontal:n { -\__enumext_listparindent_viii_dim } \ignorespaces
5125   \tl_use:N \__enumext_fake_item_indent_viii_tl
5126   \bool_if:NT \__enumext_item_starred_viii_bool
5127   {
5128     \__enumext_keyans_show_item_opt_viii:
5129   }
5130   \tl_use:N \__enumext_after_list_args_viii_tl
5131 }
5132 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
5133 {
5134   \__enumext_endminipage:
5135   \__enumext_stop_list_tag:n {keyans*}
5136   \hbox_set_end:
5137   \int_set:Nn \hbadness { 10000 }
5138   \box_use_drop:N \__enumext_item_text_viii_box
5139   \int_compare:nNnTF
5140   { \__enumext_item_column_pos_viii_int } = { \__enumext_columns_viii_int }
5141   {
5142     \par\noindent
5143     \int_zero:N \__enumext_item_column_pos_viii_int
5144   }
5145   {
5146     \skip_horizontal:N \__enumext_columns_sep_viii_dim
5147   }
5148 }

```

(End of definition for __enumext_start_item_viii:w and __enumext_stop_item_viii:.)

`__enumext_remove_extra_parsep_viii:` The implementation at this is very similar to that of the `enumext*` environment.

```

5149 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
5150 {
5151   \int_compare:nNnT
5152     {
5153       \int_mod:nn
5154         { \g__enumext_item_count_all_viii_int }
5155         { \l__enumext_columns_viii_int }
5156     }
5157     =
5158     { 0 }
5159     {
5160       \para_end:
5161       \skip_vertical:n { -\l__enumext_itemsep_viii_skip }
5162       \skip_vertical:N \c_zero_skip
5163       \int_gzero:N \g__enumext_item_count_all_viii_int
5164     }
5165   }

```

(End of definition for `__enumext_remove_extra_parsep_viii:`)

13.46 The command `\getkeyans`

`\getkeyans` The `\getkeyans` command takes a *mandatory argument* of the form $\{\langle store\ name\rangle:\langle position\rangle\}$. Retrieve a “single content” stored by `\anskey`, `\anspic*` and `\item*` and `anskey*` from *prop list* defined by `save-ans` key.

```

\__enumext_getkeyans_aux:n
\__enumext_getkeyans:nn

```

```

5166 \NewDocumentCommand \getkeyans { m }
5167 {
5168   \exp_args:Ne \__enumext_getkeyans_aux:n
5169   { \tl_to_str:e { \text_expand:n {#1} } }
5170 }

```

The internal function `__enumext_getkeyans_aux:n` is in charge of *splitting* the *mandatory argument* using “.”. If “.” is omitted it will return an error.

```

5171 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
5172 {
5173   \str_if_in:nnTF {#1} { : }
5174   {
5175     \use:e
5176     {
5177       \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
5178       { {##1} {##2} }
5179     }
5180     \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
5181   }
5182   { \msg_error:nnn { enumext } { missing-colon } {#1} }
5183 }

```

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*.

```

5184 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
5185 {
5186   \prop_if_exist:cTF { g__enumext_#1_prop }
5187   {
5188     \prop_item:cn { g__enumext_#1_prop }{#2}
5189   }
5190   {
5191     \msg_error:nnn { enumext } { undefined-storage-anskey } {#1}
5192   }
5193 }

```

(End of definition for `\getkeyans`, `__enumext_getkeyans_aux:n`, and `__enumext_getkeyans:nn`. This function is documented on page 18.)

13.47 The command `\printkeyans`

The `\printkeyans` command prints “all stored content” in the *sequence* defined by the `save-ans` key.

The first thing we will do is define a set of $\langle filtered\ keys\rangle$ with which we will control the options of the different nesting levels for the environment `enumext` and `enumext*` by storing their values in the list of tokens `\l__enumext_print_keyans_X_tl`.

The variable `\l__enumext_print_keyans_starred_tl` will have the default $\langle keys\rangle$ for `\printkeyans*` and will be set by `\setenumext[$\langle print*\rangle$]` and the variable `\l__enumext_print_keyans_vii_tl` will

have the default keys for the environment `enumext*` nested within the *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⟩]`.

```

5194 \keys_define:nn { enumext / print }
5195 {
5196   print* .code:n      = \keys_precompile:neN { enumext / enumext* }
5197                       { \__enumext_filter_save_key:n {#1} }
5198                       \l__enumext_print_keyans_starred_tl, % starred cmd
5199   print* .initial:n   = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5200                       rightmargin=0pt, listparindent=0pt, nosep, label=\arabic*.,
5201                       columns=2, first=\small, font=\small },
5202   print-1 .code:n     = \keys_precompile:neN { enumext / level-1 }
5203                       { \__enumext_filter_save_key:n {#1} }
5204                       \l__enumext_print_keyans_i_tl,
5205   print-1 .initial:n  = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5206                       rightmargin=0pt, listparindent=0pt, nosep, label=\arabic*.,
5207                       columns=2, first=\small, font=\small },
5208   print-2 .code:n     = \keys_precompile:neN { enumext / level-2 }
5209                       { \__enumext_filter_save_key:n {#1} }
5210                       \l__enumext_print_keyans_ii_tl,
5211   print-2 .initial:n  = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5212                       rightmargin=0pt, listparindent=0pt, nosep, label=(\alph*),
5213                       first=\small, font=\small },
5214   print-3 .code:n     = \keys_precompile:neN { enumext / level-3 }
5215                       { \__enumext_filter_save_key:n {#1} }
5216                       \l__enumext_print_keyans_iii_tl,
5217   print-3 .initial:n  = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5218                       rightmargin=0pt, listparindent=0pt, nosep, label=\roman*.,
5219                       first=\small, font=\small },
5220   print-4 .code:n     = \keys_precompile:neN { enumext / level-4 }
5221                       { \__enumext_filter_save_key:n {#1} }
5222                       \l__enumext_print_keyans_iv_tl,
5223   print-4 .initial:n  = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5224                       rightmargin=0pt, listparindent=0pt, nosep, label=\Alph*.,
5225                       first=\small, font=\small },
5226   print-* .code:n     = \keys_precompile:neN { enumext / enumext* }
5227                       { \__enumext_filter_save_key:n {#1} }
5228                       \l__enumext_print_keyans_vii_tl, % starred nested
5229   print-* .initial:n  = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5230                       rightmargin=0pt, listparindent=0pt, nosep, label=\arabic*.,
5231                       first=\small, font=\small },
5232 }

```

- 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`.

```

5233 \NewDocumentCommand \printkeyans { s O{} m }
5234 {
5235   \group_begin:
5236     \tl_use:N \l__enumext_print_keyans_i_tl
5237     \tl_use:N \l__enumext_print_keyans_ii_tl
5238     \tl_use:N \l__enumext_print_keyans_iii_tl
5239     \tl_use:N \l__enumext_print_keyans_iv_tl
5240     \tl_use:N \l__enumext_print_keyans_vii_tl
5241     \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
5242   \group_end:
5243 }

```

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.

```

5244 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
5245 {
5246   \seq_if_exist:cTF { g__enumext_#3_seq }
5247   {
5248     \seq_if_empty:cF { g__enumext_#3_seq }
5249     {

```

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 $\langle keys \rangle$ 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.

```

5250         \bool_if:nTF {#1}
5251         {
5252             \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
5253             {
5254                 \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
5255             }
5256             {
5257                 \tl_use:N \l__enumext_print_keyans_starred_tl
5258                 \bool_set_true:N \l__enumext_base_line_fix_bool
5259                 \bool_set_true:N \l__enumext_print_keyans_star_bool
5260                 \begin{enumext*}[#2]
5261                     \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5262                 \end{enumext*}
5263                 \bool_set_false:N \l__enumext_base_line_fix_bool
5264                 \bool_set_false:N \l__enumext_print_keyans_star_bool
5265             }
5266         }

```

Otherwise it will open the environment `enumext` passing the *optional argument* to the “first level” then map the *sequence*.

```

5267         {
5268             \begin{enumext}[#2]
5269             \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5270             \end{enumext}
5271         }
5272     }
5273 }
5274 {
5275     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
5276 }
5277 }

```

(End of definition for `\printkeyans` and `__enumext_printkeyans:nnn`. This function is documented on page 19.)

13.48 The command `\setenumext`

The command `\setenumext` will be in charge of managing the $\langle keys \rangle$ passed to all environments and to the `\printkeyans` command. We must take precautions with the `enumext*` environment and “first level” of the `enumext` environment so as not to capture $\langle keys \rangle$ that complicate us.

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

```

5278 \cs_new:Npn \__enumext_filter_first_level:n #1
5279 {
5280     \use:e
5281     {
5282         \keyval_parse:NNn
5283         \__enumext_filter_first_level_key:n
5284         \__enumext_filter_first_level_pair:nn {#1}
5285     }
5286 }

```

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

```

5287 \cs_new:Npn \__enumext_filter_first_level_key:n #1
5288 {
5289     \str_case:nnF {#1}
5290     {
5291         { resume } {}
5292         { resume* } {}
5293     }
5294     { , { \exp_not:n {#1} } }
5295 }

```

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.

```

5296 \cs_new:Npn \__enumext_filter_first_level_pair:nn #1#2
5297 {
5298   \str_case:nnF {#1}
5299   {
5300     { series } {}
5301     { resume } {}
5302     { save-ans } {}
5303   }
5304   { , { \exp_not:n {#1} } } = { \exp_not:n {#2} } }
5305 }

```

(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`.

```

5306 \keys_define:nn { enumext / meta-families }
5307 {
5308   enumext-1 .code:n =
5309   {
5310     \keys_set:ne { enumext / level-1 }
5311     {
5312       \__enumext_filter_first_level:n {#1}
5313     }
5314   } ,
5315   enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
5316   enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
5317   enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
5318   keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} } ,
5319   enumext* .code:n =
5320   {
5321     \keys_set:ne { enumext / enumext* }
5322     {
5323       \__enumext_filter_first_level:n {#1}
5324     }
5325   } ,
5326   keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
5327   print* .code:n = { \keys_set:nn { enumext / print } { print* = {#1} } } ,
5328   print-1 .code:n = { \keys_set:nn { enumext / print } { print-1 = {#1} } } ,
5329   print-2 .code:n = { \keys_set:nn { enumext / print } { print-2 = {#1} } } ,
5330   print-3 .code:n = { \keys_set:nn { enumext / print } { print-3 = {#1} } } ,
5331   print-4 .code:n = { \keys_set:nn { enumext / print } { print-4 = {#1} } } ,
5332   print-* .code:n = { \keys_set:nn { enumext / print } { print-* = {#1} } } ,
5333   unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
5334 }

```

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

```

5335 \seq_const_from_clist:Nn \c__enumext_all_families_seq
5336 {
5337   enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
5338   keyans*, print-1, print-2, print-3, print-4, print-*, print*,
5339 }

```

`\setenumext`

Now we define the user command `\setenumext`.

```

\__enumext_set_parse:n
\__enumext_set_error:nn
5340 \NewDocumentCommand \setenumext { 0{enumext,1} +m }
5341 {
5342   \seq_clear:N \l__enumext_setkey_tmpa_seq
5343   \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
5344   \int_set:Nn \l__enumext_setkey_tmpa_int
5345   {
5346     \seq_count:N \l__enumext_setkey_tmpb_seq
5347   }
5348   \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
5349   {
5350     \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
5351     \seq_map_function:NN \l__enumext_setkey_tmpb_seq \__enumext_set_parse:n
5352     \seq_set_map:e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
5353     {
5354       \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
5355     }

```

```

5356     }
5357     {
5358         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
5359     }
5360     \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
5361     { \seq_map_inline:Nn \c__enumext_all_families_seq }
5362     { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
5363     {
5364         \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
5365     }
5366 }

```

Internal functions used by the `\setenumext` command.

```

5367 \cs_new_protected:Npn \__enumext_set_parse:n #1
5368 {
5369     \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
5370     \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
5371     { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
5372     \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
5373     {
5374         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
5375         { \tl_trim_spaces:n {#1} }
5376     }
5377     { \__enumext_set_error:nn {#1} { } }
5378 }
5379 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
5380 { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }

```

(End of definition for `\setenumext`, `__enumext_set_parse:n`, and `__enumext_set_error:nn`. This function is documented on page 6.)

13.49 The command `\setenumextmeta`

The command `\setenumextmeta` will be responsible for adding new “meta-keys” for the `enumext` and `enumext*` environments. The implementation code was given by Jonathan P. Spratte (@Skillmon) answer in [Add .meta key to existing keys \(l3keys\)](#).

`\setenumextmeta`

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
5381 \prop_const_from_keyval:Nn \c__enumext_meta_paths_prop
5382 {
5383     {enumext,1} = level-1,
5384     {enumext,2} = level-2,
5385     {enumext,3} = level-3,
5386     {enumext,4} = level-4,
5387     {enumext*} = enumext*
5388 }

```

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

```

5389 \NewDocumentCommand \setenumextmeta { s O{enumext,1} m +m }
5390 {
5391     \str_if_eq:eeTF { \tl_trim_spaces:n {#3} } { unknown }
5392     { \msg_error:nn { enumext } { prohibited-unknown } }
5393     {
5394         \bool_if:nTF {#1}
5395         {
5396             \int_step_inline:nn { 4 }
5397             { \__enumext_add_meta_key:nnn { enumext, ##1 } {#3} {#4} }
5398             \__enumext_add_meta_key:nnn { enumext* } {#3} {#4}
5399         }
5400         { \__enumext_add_meta_key:nnn {#2} {#3} {#4} }
5401     }
5402 }

```

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

```

5403 \cs_new_protected:Npn \__enumext_add_meta_key:nnn #1
5404 {
5405     \tl_set:Nn \l__enumext_meta_path_tl {#1}
5406     \tl_replace_all:Nnn \l__enumext_meta_path_tl {~} {}
5407     \prop_get:NVNTF
5408     \c__enumext_meta_paths_prop \l__enumext_meta_path_tl \l__enumext_meta_path_tl
5409     { \__enumext_def_meta_key:Vnn \l__enumext_meta_path_tl }
5410 }

```

```

5411         \msg_error:nnn { enumext } { unknown-set } {#1}
5412         \use_none:nn
5413     }
5414 }
5415 \cs_new_protected:Npn \__enumext_def_meta_key:nnn #1#2#3
5416 {
5417     \bool_lazy_or:nnTF
5418     { \keys_if_exist_p:nn { enumext / #1 } {#2} }
5419     { \keys_if_exist_p:nn { enumext / enumext* } {#2} }
5420     { \msg_error:nnn { enumext } { already-defined } {#2} }
5421     {
5422         \keys_define:nn { enumext / #1 }
5423         {
5424             #2 .meta:n = {#3},
5425             #2 .value_forbidden:n = true
5426         }
5427     }
5428 }
5429 \cs_generate_variant:Nn \__enumext_def_meta_key:nnn { V }

```

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

13.50 The command `\foreachkeyans`

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

`\foreachkeyans`

`__enumext_parse_foreach_keys:nn`

`__enumext_parse_foreach_keys:n`

`__enumext_foreach_keyans:nn`

`__enumext_foreach_add_body:n`

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

```

5430 \keys_define:nn { enumext / foreach }
5431 {
5432     before .tl_set:N = \__enumext_foreach_before_tl,
5433     before .value_required:n = true,
5434     after .tl_set:N = \__enumext_foreach_after_tl,
5435     after .value_required:n = true,
5436     start .int_set:N = \__enumext_foreach_start_int,
5437     start .value_required:n = true,
5438     stop .int_set:N = \__enumext_foreach_stop_int,
5439     stop .value_required:n = true,
5440     step .int_set:N = \__enumext_foreach_step_int,
5441     step .value_required:n = true,
5442     wrapper .cs_set_protected:Np = \__enumext_foreach_wrapper:n #1,
5443     wrapper .value_required:n = true,
5444     sep .tl_set:N = \__enumext_foreach_sep_tl,
5445     sep .value_required:n = true,
5446     unknown .code:n = { \__enumext_parse_foreach_keys:n {#1} }
5447 }
5448 \keys_precompile:nnN { enumext / foreach }
5449 {
5450     before={},after={},start=1,step=1,stop=0,wrapper=#1,sep={; }
5451 }
5452 \l__enumext_foreach_default_keys_tl

```

Functions for handling unknown *⟨keys⟩*.

```

5453 \cs_new_protected:Npn \__enumext_parse_foreach_keys:nn #1#2
5454 {
5455     \tl_if_blank:nTF {#2}
5456     {
5457         \msg_error:nnn { enumext } { for-key-unknown } {#1}
5458     }
5459     {
5460         \msg_error:nnnn { enumext } { for-key-value-unknown } {#1} {#2}
5461     }
5462 }
5463 \cs_new_protected:Npn \__enumext_parse_foreach_keys:n #1
5464 {
5465     \exp_args:NV \__enumext_parse_foreach_keys:nn \l_keys_key_str {#1}
5466 }

```

We create the command.

```

5467 \NewDocumentCommand \foreachkeyans { +0{} m }
5468 {

```

```

5469   \__enumext_foreach_keyans:nn {#1} {#2}
5470 }

```

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

```

5471 \cs_new_protected:Npn \__enumext_foreach_keyans:nn #1 #2
5472 {
5473   \tl_use:N \l__enumext_foreach_default_keys_tl
5474   \keys_set:nn { enumext / foreach } {#1}
5475   \tl_set:Nn \l__enumext_foreach_name_prop_tl {#2}
5476   \prop_if_exist:cF { g__enumext_#2_prop }
5477   {
5478     \msg_error:nnn { enumext } { undefined-storage-anskey } {#2}
5479   }
5480   \int_compare:nNt { \l__enumext_foreach_stop_int } = { 0 }
5481   {
5482     \int_set:Nn \l__enumext_foreach_stop_int
5483     { \prop_count:c { g__enumext_#2_prop } }
5484   }
5485   \seq_clear:N \l__enumext_foreach_print_seq
5486   \int_step_function:nnnN
5487   { \l__enumext_foreach_start_int }
5488   { \l__enumext_foreach_step_int }
5489   { \l__enumext_foreach_stop_int }
5490   \__enumext_foreach_add_body:n
5491   \seq_use:NV \l__enumext_foreach_print_seq \l__enumext_foreach_sep_tl
5492 }
5493 \cs_new_protected:Npn \__enumext_foreach_add_body:n #1
5494 {
5495   \seq_put_right:Ne \l__enumext_foreach_print_seq
5496   {
5497     \exp_not:V \l__enumext_foreach_before_tl
5498     \__enumext_foreach_wrapper:n
5499     {
5500       \prop_item:cn { g__enumext_ \l__enumext_foreach_name_prop_tl _prop }{#1}
5501     }
5502     \exp_not:V \l__enumext_foreach_after_tl
5503   }
5504 }

```

(End of definition for `\foreachkeyans` and others. This function is documented on page 18.)

13.51 Messages

Message used by package-load for `multicol` and `hyperref` packages.

```

5505 \msg_new:nnn { enumext } { package-load }
5506 {
5507   The~'#1'~package~is~already~loaded.
5508 }
5509 \msg_new:nnn { enumext } { package-not-load }
5510 {
5511   The~'#1'~package~will~be~loaded~as~a~dependency.
5512 }
5513 \msg_new:nnn { enumext } { package-load-foot }
5514 {
5515   The~'#1'~package~is~loaded~with~the~option~'#2'.
5516 }

```

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

```

5517 \msg_new:nnn { enumext } { counters }
5518 {
5519   The~counter~'#1'~is~already~defined~by~some~\\
5520   package~or~macro,~it~cannot~be~continued.
5521 }

```

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

```

5522 \msg_new:nnn { enumext } { unknown-choice }
5523 {
5524   The~value~'#3'~for~'#1'~key~is~invalid~use~('#2').
5525 }

```

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

```

5526 \msg_new:nnnn { enumext } { anskey-env-error }

```

```

5527 {
5528     The~environment~'#1'~is~reserved~by ~\\
5529     'enumext'~package,~It~is~already~defined.
5530 }
5531 {
5532     The~environment~'#1'~is~defined~internally ~
5533     for~the~'save-ans'~key~with~save-ans~key~active.~See~documentation.\\
5534 }
5535 \msg_new:nnn { enumext } { anskey-env-nested }
5536 {
5537     The~#1~'#2'~can't~be~nested~\msg_line_context:.
5538 }

```

Message used in the creation of *prop list* by enumext package.

```

5539 \msg_new:nnn { enumext } { store-prop }
5540 {
5541     *~Package~enumext:~Creating ~
5542     \c_backslash_str g__enumext_#1_prop~\msg_line_context:.
5543 }
5544 \msg_new:nnn { enumext } { store-seq }
5545 {
5546     *~Package~enumext:~Creating ~
5547     \c_backslash_str g__enumext_#1_seq~\msg_line_context:.
5548 }
5549 \msg_new:nnn { enumext } { store-int }
5550 {
5551     *~Package~enumext:~Creating ~
5552     \c_backslash_str g__enumext_resume_#1_int~\msg_line_context:.
5553 }
5554 \msg_new:nnn { enumext } { prop-seq-int-hook }
5555 {
5556     *~Package~enumext:~Elements~in ~
5557     \c_backslash_str g__enumext_#1_prop~::~#2.\\
5558     *~Package~enumext:~Elements~in ~
5559     \c_backslash_str g__enumext_#1_seq~::~#3.\\
5560     *~Package~enumext:~Value~off ~
5561     \c_backslash_str g__enumext_resume_#1_int~::~#4.
5562 }
5563 \msg_new:nnn { enumext } { item-answer-hook }
5564 {
5565     *~Package~enumext:~Value~off ~
5566     \c_backslash_str g__enumext_item_number_int~::~#1.\\
5567     *~Package~enumext:~Value~off ~
5568     \c_backslash_str g__enumext_item_anskey_int~::~#2.\\
5569     *~Package~enumext:~Difference~item_number_int~::~item_anskey_int~::~#3.
5570 }

```

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

```

5571 \msg_new:nnn { enumext } { invalid-key }
5572 {
5573     The~key~'#1'~is~not~know~the~level~#2.
5574 }
5575 \msg_new:nnn { enumext } { unknown-key-family }
5576 {
5577     Unknown~key~family~`\l_keys_key_str'~for~enumext.
5578 }

```

Messages used in length calculation.

```

5579 \msg_new:nnn { enumext } { width-negative }
5580 {
5581     Ignoring~negative~value~'#1=#2'~\msg_line_context:\\.
5582     The~key~'#1'~ accepts~values ~>=~0pt.
5583 }
5584 \msg_new:nnn { enumext } { width-zero }
5585 {
5586     Invalid~'#1=#2'~\msg_line_context:\\.
5587     The~key~'#1'~ accepts~values ~>~0pt.
5588 }

```

Messages used by show-length key in enumext.

```

5589 \msg_new:nnn { enumext } { list-lengths }
5590 {
5591     ****~Lengths~used~by~'enumext'~level~'#2'~\msg_line_context:~\c_space_tl ****\\

```



```

5592     \__enumext_show_length:nnn { dim } { labelsep } {#1}
5593     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
5594     \__enumext_show_length:nnn { dim } { itemindent } {#1}
5595     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
5596     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
5597     \__enumext_show_length:nnn { dim } { listparindent } {#1}
5598     \__enumext_show_length:nnn { skip } { topsep } {#1}
5599     \__enumext_show_length:nnn { skip } { parsep } {#1}
5600     \__enumext_show_length:nnn { skip } { partopsep } {#1}
5601     \__enumext_show_length:nnn { skip } { itemsep } {#1}
5602     *****
5603 }

```

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

```

5604 \msg_new:nnn { enumext } { list-lengths-not-nested }
5605 {
5606     ****~Lengths~used~by~'#2'~environment~\msg_line_context:~\c_space_tl ****\\
5607     \__enumext_show_length:nnn { dim } { labelsep } {#1}
5608     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
5609     \__enumext_show_length:nnn { dim } { itemindent } {#1}
5610     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
5611     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
5612     \__enumext_show_length:nnn { dim } { listparindent } {#1}
5613     \__enumext_show_length:nnn { skip } { topsep } {#1}
5614     \__enumext_show_length:nnn { skip } { parsep } {#1}
5615     \__enumext_show_length:nnn { skip } { partopsep } {#1}
5616     \__enumext_show_length:nnn { skip } { itemsep } {#1}
5617     *****
5618 }

```

Messages used by `ref` key.

```

5619 \msg_new:nnn { enumext } { key-ref-empty }
5620 {
5621     Key~'ref'~need~a~value~in~'#1'~ \msg_line_context:.
5622 }

```

Messages used by `save-ans` key.

```

5623 \msg_new:nnn { enumext } { save-ans-empty }
5624 {
5625     Key~'save-ans'~need~a~value~in~'#1'~ \msg_line_context:.
5626 }
5627 \msg_new:nnn { enumext } { save-ans-log }
5628 {
5629     *~Package~enumext:~Start~#1\c_space_tl with~save-ans=#2~\msg_line_context:.
5630 }
5631 \msg_new:nnn { enumext } { save-ans-log-hook }
5632 {
5633     *~Package~enumext:~Stop~#1\c_space_tl with~save-ans=#2~\msg_line_context:.
5634 }
5635 \msg_new:nnn { enumext } { save-ans-hook }
5636 {
5637     Stop~storing~for~'save-ans=#1'~\msg_line_context:.
5638 }

```

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

```

5639 \msg_new:nnn { enumext } { need-save-ans }
5640 {
5641     Key~'#1'~ works~only~with~the~'save-ans'~key~in~'#2'~ \msg_line_context:.
5642 }
5643 \msg_new:nnn { enumext } { items-same-answer }
5644 {
5645     *****\\
5646     *~Package~enumext:~Checking~answers~in~'#1' ~
5647     for~\c_left_brace_str #2 \c_right_brace_str\\
5648     *~started~#3~and~close~\msg_line_context: : ~
5649     'OK',~all~items~with~answer.\\
5650     *****
5651 }
5652 \msg_new:nnn { enumext } { item-greater-answer }
5653 {
5654     Checking~answers~in~'#1'~for~\c_left_brace_str #2 \c_right_brace_str\\
5655     started~#3~and~close~\msg_line_context: : ~'NOT~OK'\\
5656     Items~>~Answers.

```

```

5657     }
5658     \msg_new:nnn { enumext } { item-less-answer }
5659     {
5660       Checking~answers~in~'#1'~for~\c_left_brace_str #2 \c_right_brace_str\\
5661       started~#3~and~close~\msg_line_context: : ~'NOT~OK'\\
5662       Items~<~Answers.
5663     }

```

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

```

5664 \msg_new:nnn { enumext } { missing-starred }
5665 {
5666   Missing~'\c_backslash_str #1'~#2.
5667 }
5668 \msg_new:nnn { enumext } { many-starred }
5669 {
5670   Many~'\c_backslash_str #1'~#2.
5671 }

```

Messages used by `\printkeyans*` command.

```

5672 \msg_new:nnn { enumext } { print-starred }
5673 {
5674   \c_backslash_str printkeyans*:~ The~sequence~'#1'~already~contains ~
5675   #2~environment~ \msg_line_context:.
5676 }

```

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

```

5677 \msg_new:nnn { enumext } { list-too-deep }
5678 {
5679   Too~deep~nesting ~for~'enumext'~\msg_line_context:~ \\
5680   The~maximum ~level ~of ~nesting ~is~4.
5681 }

```

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

```

5682 \msg_new:nnn { enumext } { anskey-unnumber-item }
5683 {
5684   Can't~store~with~a~unnumbered~\c_backslash_str item~\msg_line_context:.
5685 }
5686 \msg_new:nnn { enumext } { anskey-already-stored }
5687 {
5688   Content~already~stored~for~this~\c_backslash_str item~\msg_line_context:.
5689 }
5690 \msg_new:nnn { enumext } { anskey-empty-arg }
5691 {
5692   Can't~store~empty~content~\msg_line_context:.
5693 }
5694 \msg_new:nnn { enumext } { anskey-wrong-place }
5695 {
5696   Wrong~place~for~command~'\c_backslash_str #1'~\msg_line_context:~ \\
5697   '\c_backslash_str #1'~works~in~the~environment~'#2'.
5698 }
5699 \msg_new:nnn { enumext } { anskey-nested }
5700 {
5701   The~command~\c_backslash_str anskey~ can't~be~nested~\msg_line_context:.
5702 }
5703 \msg_new:nnn { enumext } { anskey-math-mode }
5704 {
5705   #1~can't~work~in~math~mode~\msg_line_context:.
5706 }
5707 \msg_new:nnn { enumext } { anskey-env-wrong }
5708 {
5709   The~environment~anskey*~cannot~use~in~'#1'~\msg_line_context:.
5710 }
5711 \msg_new:nnn { enumext } { anspic-wrong-place }
5712 {
5713   Wrong~place~for~command~'\c_backslash_str #1'~\msg_line_context:~ \\
5714   '\c_backslash_str #1'~works~in~the~environment~'#2'.
5715 }
5716 \msg_new:nnn { enumext } { command-wrong-place }
5717 {
5718   Wrong~place~for~command~'\c_backslash_str #1'~\msg_line_context:~ \\
5719   '\c_backslash_str #1'~works~outside~the~environment~'#2'.
5720 }
5721 \msg_new:nnnn { enumext } { anskey-env-key-unknown }

```

```

5722 {
5723     The~key~'#1'~is~unknown~by~environment~
5724     'anskey* '~and~is~being~ignored.
5725 }
5726 {
5727     The~environment~'anskey* '~does~not~have~a~key~called ~'#1'.\\
5728     Check~that~you~have~spelled~the~key~name~correctly.
5729 }
5730 \msg_new:nnnn { enumext } { anskey-env-key-value-unknown }
5731 {
5732     The~key~'#1=#2'~is~unknown~by~environment ~
5733     'anskey* '~and~is~being~ignored.
5734 }
5735 {
5736     The~environment~'anskey* '~does~not~have~a~key~called ~'#1'.\\
5737     Check~that~you~have~spelled~the~key~name~correctly.
5738 }
5739 \msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
5740 { The~key~'#1'~is~unknown~by~'\c_backslash_str anskey'~and~is~being~ignored.}
5741 {
5742     The~command ~'\c_backslash_str anskey'~does~not~have~a~key~called ~'#1'.\\
5743     Check~that~you~have~spelled~the~key~name~correctly.
5744 }
5745 \msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
5746 { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str anskey'~and~is~being~ignored. }
5747 {
5748     The~command~'\c_backslash_str anskey'~does~not~have~a~key~called ~'#1'.\\
5749     Check~that~you~have~spelled~the~key~name~correctly.
5750 }
5751 \msg_new:nnn { enumext } { overwrite-file }
5752 {
5753     Overwriting~file~'#1'.
5754 }
5755 \msg_new:nnn { enumext } { writing-file }
5756 {
5757     Writing~file~'#1'.
5758 }
5759 \msg_new:nnn { enumext } { not-writing }
5760 {
5761     File~'#1'~already~exists.~Not~writing.
5762 }

```

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

```

5763 \msg_new:nnn { enumext } { keyans-nested }
5764 {
5765     The~environment~'keyans'~can't~be ~nested ~\msg_line_context:.
5766 }
5767 \msg_new:nnn { enumext } { keyans-wrong-level }
5768 {
5769     Wrong~level~position~for~'keyans'~\msg_line_context:.~ \\
5770     The~environment~'keyans'~can~only~be~in~the~first~level.
5771 }
5772 \msg_new:nnn { enumext } { wrong-place }
5773 {
5774     Wrong~place~for~'#1'~environment ~\msg_line_context:.~ \\
5775     '#1'~is~only~found~with~'#2'~ in ~ 'enumext.
5776 }
5777 \msg_new:nnn { enumext } { keyanspic-nested }
5778 {
5779     The~environment~'keyanspic'~can't~be ~nested~ \msg_line_context:.~.
5780 }
5781 \msg_new:nnn { enumext } { keyanspic-wrong-level }
5782 {
5783     Wrong~level~position~for~'keyanspic'~\msg_line_context:.~ \\
5784     The~environment~'keyans'~can~only~be~in~the~first~level.
5785 }
5786 \msg_new:nnn { enumext } { keyanspic-item-cmd }
5787 {
5788     Can't~use ~\c_backslash_str item~in~keyanspic~\msg_line_context:.
5789 }
5790 \msg_new:nnnn { enumext } { keyans-unknown-key }
5791 {

```

```

5792     The~key~'#1'~is~unknown~by~environment~
5793     '\l__enumext_envir_name_tl'~and~is~being~ignored.
5794 }
5795 {
5796     The~environment~'\l__enumext_envir_name_tl'~does~not
5797     ~have~a~key~called ~'#1'.\\
5798     Check~that~you~have~spelled~the~key~name~correctly.
5799 }
5800 \msg_new:nnnn { enumext } { keyans-unknown-key-value }
5801 {
5802     The~key~'#1=#2'~is~unknown~by~environment ~
5803     '\l__enumext_envir_name_tl'~and~is~being~ignored.
5804 }
5805 {
5806     The~environment~'\l__enumext_envir_name_tl'~does~not
5807     ~have~a~key~called ~'#1'.\\
5808     Check~that~you~have~spelled~the~key~name~correctly.
5809 }

```

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

```

5810 \msg_new:nnnn { enumext } { starred-unknown-key }
5811 {
5812     The~key~'#1'~is~unknown~by~environment~
5813     '\l__enumext_envir_name_tl'~and~is~being~ignored.
5814 }
5815 {
5816     The~environment~'\l__enumext_envir_name_tl'~does~not
5817     ~have~a~key~called ~'#1'.\\
5818     Check~that~you~have~spelled~the~key~name~correctly.
5819 }
5820 \msg_new:nnnn { enumext } { starred-unknown-key-value }
5821 {
5822     The~key~'#1=#2'~is~unknown~by~environment ~
5823     '\l__enumext_envir_name_tl'~and~is~being~ignored.
5824 }
5825 {
5826     The~environment~'\l__enumext_envir_name_tl'~does~not
5827     ~have~a~key~called ~'#1'.\\
5828     Check~that~you~have~spelled~the~key~name~correctly.
5829 }

```

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

```

5830 \msg_new:nnnn { enumext } { standar-unknown-key }
5831 {
5832     The~key~'#1'~is~unknown~by~environment~'\l__enumext_envir_name_tl' \c_space_tl
5833     ~on~level~\int_use:N \l__enumext_level_int \c_space_tl and~is~being~ignored.
5834 }
5835 {
5836     The~environment~'\l__enumext_envir_name_tl'~does~not
5837     ~have~a~key~called ~'#1'~on~level~\int_use:N \l__enumext_level_int.\\
5838     Check~that~you~have~spelled~the~key~name~correctly.
5839 }
5840 \msg_new:nnnn { enumext } { standar-unknown-key-value }
5841 {
5842     The~key~'#1=#2'~is~unknown~by~environment~'\l__enumext_envir_name_tl' \c_space_tl
5843     ~on~level~\int_use:N \l__enumext_level_int \c_space_tl and~is~being~ignored.
5844 }
5845 {
5846     The~environment~'\l__enumext_envir_name_tl'~does~not
5847     ~have~a~key~called ~'#1'~on~level~\int_use:N \l__enumext_level_int.\\
5848     Check~that~you~have~spelled~the~key~name~correctly.
5849 }

```

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

```

5850 \msg_new:nnnn { enumext } { for-key-unknown }
5851 { The~key~'#1'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored.}
5852 {
5853     The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5854     Check~that~you~have~spelled~the~key~name~correctly.
5855 }
5856 \msg_new:nnnn { enumext } { for-key-value-unknown }
5857 { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored. }

```

```

5858 {
5859     The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5860     Check~that~you~have~spelled~the~key~name~correctly.
5861 }

```

Messages used by `\getkeyans` command.

```

5862 \msg_new:nnn { enumext } { undefined-storage-anskey }
5863 {
5864     Storage~named~'#1'~is~not~defined~\msg_line_context:.
5865 }

```

Messages used by `\miniright` command.

```

5866 \msg_new:nnn { enumext } { missing-miniright }
5867 {
5868     Missing~'\c_backslash_str miniright'~in~\msg_line_context:.\\
5869     The~key~'mini-env'~need~'\c_backslash_str miniright'.
5870 }
5871 \msg_new:nnn { enumext } { wrong-miniright-place }
5872 {
5873     Wrong~place~for~'\c_backslash_str miniright'~\msg_line_context:.~ \\
5874     Works~in~'enumext'~and~'keyans'~with~key~'mini-env'.
5875 }
5876 \msg_new:nnn { enumext } { wrong-miniright-use }
5877 {
5878     Wrong~use~for~'\c_backslash_str miniright'~\msg_line_context:.~ \\
5879     '\c_backslash_str miniright'~need~a~key~'mini-env'.
5880 }
5881 \msg_new:nnn { enumext } { wrong-miniright-starred }
5882 {
5883     Can't~use ~\c_backslash_str miniright~in~starred~environments~\msg_line_context:.
5884 }
5885 \msg_new:nnn { enumext } { many-miniright-used }
5886 {
5887     Can't~use ~\c_backslash_str miniright~more~than~once~ \msg_line_context:.
5888 }

```

Messages used by `\setenumextmeta` command.

```

5889 \msg_new:nnn { enumext } { unknown-set }
5890 {
5891     Argument~[#1]~is~unknown~by~ \c_backslash_str setenumextmeta~\msg_line_context:.
5892 }
5893 \msg_new:nnn { enumext } { already-defined }
5894 {
5895     The~key~'#1'~is~already~defined~\msg_line_context:.
5896 }
5897 \msg_new:nnn { enumext } { prohibited-unknown }
5898 {
5899     The~name~'unknown'~can't~be~chosen~ for~a~meta~key~\msg_line_context:.
5900 }

```

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

```

5901 \msg_new:nnn { enumext } { nested }
5902 {
5903     The~environment~\l__enumext_envir_name_tl \c_space_tl can't~be~nested~\msg_line_context:.
5904 }
5905 \msg_new:nnn { enumext } { nested-horizontal }
5906 {
5907     The~environment~\l__enumext_envir_name_tl \c_space_tl can't~be~nested~in~'#1'~ \msg_line_cont
5908 }
5909 \msg_new:nnn { enumext } { item-joined }
5910 {
5911     Items~joined~( #1 )~>~#2 ~columns ~\msg_line_context:.
5912 }
5913 \msg_new:nnn { enumext } { item-joined-columns }
5914 {
5915     Not~space~to~join~items~( #1 )~>~#2 ~\msg_line_context:.
5916 }

```

13.52 Finish package

Finish package implementation.

```

5917 \file_input_stop:
5918 </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>	217
<code>\-</code>	217
<code>\\</code>	225, 4296, 4299, 5519, 5528, 5533, 5557, 5559, 5566, 5568, 5581, 5586, 5591, 5606, 5645, 5647, 5649, 5654, 5655, 5660, 5661, 5679, 5696, 5713, 5718, 5727, 5736, 5742, 5748, 5769, 5774, 5783, 5797, 5807, 5817, 5827, 5837, 5847, 5853, 5859, 5868, 5873, 5878
A	
<code>above</code>	<u>1697</u>
<code>above*</code>	<u>1697</u>
<code>\addvspace</code>	1264, 1292, 1335, 1338, 1506, 1509, 1606, 1612, 1650, 1656, 1677, 1683, 3711, 3880, 3898, 4181, 4185, 4544, 4559, 4605, 4619
<code>after</code>	<u>1094</u>
<code>align</code>	<u>651</u>
<code>\Alph</code>	42, 47, 48
<code>\Alph</code>	593, 721, 765, 825, 5224
<code>\alph</code>	42, 47, 48
<code>\alph</code>	594, 719, 5212
<code>\anskey</code>	13, 82, 84, <u>2756</u>
<code>anskey*</code>	14, <u>2886</u>
<code>\anspic</code>	17, 110, 113, <u>4195</u>
<code>\anspic*</code>	75
<code>\arabic</code>	42
<code>\arabic</code>	592, 718, 764, 5200, 5206, 5230
B	
<code>base-fix</code>	<u>952</u>
<code>\baselineskip</code>	<u>56</u>
<code>\baselineskip</code>	968, 979
<code>before</code>	<u>1094</u>
<code>before*</code>	<u>1094</u>
<code>below</code>	<u>1697</u>
<code>below*</code>	<u>1697</u>
bool commands:	
<code>\bool_gset_false:N</code>	336, 337, 338, 4561, 4565, 4621
<code>\bool_gset_true:N</code>	246, 256, 1197, 2189, 2195, 4530, 4562, 4594, 4622
<code>\bool_if:NTF</code>	386, 396, 413, 487, 494, 503, 510, 524, 537, 1719, 1733, 1746, 1757, 1768, 1779, 1790, 1801, 1850, 1867, 1872, 1880, 1907, 1945, 1950, 1957, 1961, 1983, 1988, 1996, 2003, 2034, 2042, 2134, 2377, 2387, 2467, 2491, 2498, 2522, 2620, 2642, 2682, 2706, 2710, 2760, 2779, 2803, 2855, 2859, 2889, 2907, 2926, 2942, 2965, 2996, 3011, 3083, 3199, 3233, 3269, 3285, 3306, 3445, 3466, 3512, 3553, 3563, 3595, 3600, 3666, 3692, 3742, 3808, 3863, 3888, 4114, 4179, 4197, 4216, 4267, 4294, 4523, 4539, 4545, 4588, 4602, 4606, 4696, 4706, 4794, 4800, 4807, 4823, 4917, 4927, 5041, 5048, 5079, 5095, 5126
<code>\bool_if:nTF</code>	1657, 1684, 3255, 3424, 4237, 5250, 5394
<code>\bool_if_p:N</code>	265, 279, 962, 963, 975, 976, 1629, 2014, 2015, 2023, 2024, 2147, 2173, 2186, 2187, 2192, 2193, 2555, 2565, 2577, 2592, 2593, 2627, 2668, 2669, 3070, 3071, 3100, 3101, 3113, 3114, 3154, 3155, 3174, 3175, 3458, 3459, 3460, 3639, 3641, 3652, 5070, 5071, 5072
<code>\bool_lazy_all:nTF</code>	263, 277, 960, 2145, 2171, 2553, 2562, 2575, 2590, 3152, 3172, 3456, 3637, 3650, 5068
<code>\bool_lazy_and:nnTF</code>	242, 252, 974, 1621, 1628, 2013, 2022, 2185, 2191, 2626, 2633, 2667, 3069
<code>\bool_lazy_or:nnTF</code>	2075, 2082, 3099, 3112, 5417
<code>\bool_new:N</code>	22, 23, 24, 25, 26, 27, 28, 49, 58, 82, 87, 88, 93, 94, 97, 104, 119, 131, 132, 139, 145, 146, 148, 152, 154, 155, 172, 184, 186
<code>\bool_not_p:n</code>	243, 253, 964, 1630, 2564, 2628, 2634, 3640, 3653
<code>\bool_set_eq:NN</code>	3208, 3405, 4747, 4992
<code>\bool_set_false:N</code>	393, 986, 2119, 2120, 2152, 2157, 2161, 2165, 2178, 3433, 3614, 3759, 3816, 3903, 4050, 4111, 4257, 4665, 4691, 4744, 4933, 4988, 4989, 5263, 5264
<code>\bool_set_true:N</code>	270, 284, 379, 382, 644, 1001, 1703, 1708, 1970, 2092, 2093, 2409, 2417, 2830, 3202, 3204, 3236, 3238, 3401, 3412, 3426, 3576, 3613, 3646, 3659, 3732, 3813, 3840, 4047, 4239, 4240, 4512, 4577, 4664, 4751, 4758, 4759, 4803, 4931, 4996, 5003, 5004, 5005, 5258, 5259
box commands:	
<code>\box_dp:N</code>	1552, 1553, 1556, 1563, 1576, 1584, 1590, 1598, 4125, 4131, 4181, 4278
<code>\box_ht:N</code>	1335, 1338, 1349, 1350, 1361, 1363, 1378, 1381, 1389, 1390, 1401, 1403, 1418, 1421, 1428, 1429, 1440, 1442, 1457, 1460, 1506, 1509, 1517, 1518, 1526, 1527, 1539, 1541
<code>\box_ht_plus_dp:N</code>	4120, 4189, 4225
<code>\box_new:N</code>	55, 141, 142, 179, 185
<code>\box_use_drop:N</code>	4556, 4617, 4859, 5138
<code>\box_wd:N</code>	600
<code>break-col</code>	<u>2726</u> , <u>2812</u>
C	
<code>\c</code>	859, 861, 873, 875
<code>\centering</code>	1659, 1686, 4322, 4549, 4610
<code>check-ans</code>	<u>2111</u>
Document class:	
<code>article</code>	49
clist commands:	
<code>\clist_const:Nn</code>	191
<code>\clist_map_function:nN</code>	4305
<code>\clist_map_inline:Nn</code>	650, 907, 1093, 1108, 1189, 1713
<code>\clist_map_inline:nn</code>	36, 45, 63, 71, 84, 96, 134, 163, 190, 628, 681, 701, 1006, 1027, 1203, 1819, 2059, 2125, 2304, 2374, 2406, 2550, 3005, 3327, 3342, 3382, 3541, 3544, 3571, 3583, 3586, 3605, 5370
<code>\columnbreak</code>	82
<code>\columnbreak</code>	2630
<code>columns</code>	<u>1173</u>
<code>columns-sep</code>	<u>1173</u>
<code>\columnsep</code>	103
<code>\columnsep</code>	3687, 3861
<code>\columnseprule</code>	103
<code>\columnseprule</code>	3690, 3862
Commands provide by enumext :	
<code>\anskey</code>	31, 32, 71, 72, 77–81, 83, 84, 89, 102, 103, 122, 132, 133, 141
<code>\anspic*</code>	31, 33, 75, 78, 89, 90, 113, 132, 133
<code>\anspic</code>	32, 79, 110, 113, 141

<code>\foreachkeyans</code>	137, 143	159, 170, 619, 629, 651, 686, 702, 748, 882, 908, 988,
<code>\getkeyans</code>	78, 132, 144	1011, 1085, 1094, 1173, 1190, 1697, 1808, 2051, 2111,
<code>\item*</code> 31, 33, 75, 78, 79, 89, 90, 93, 97, 124, 129, 130, 132,		2270, 2305, 2393, 2543, 2998, 3316, 3332, 3375, 3532,
133		3572
<code>\item</code>	93, 97, 117, 123, 125, 128, 129	<code>\cs_to_str:N</code>
<code>\miniright</code>	31, 54, 62, 63, 104, 144	589, 612
<code>\printkeyans*</code>	132	
<code>\printkeyans</code>	32, 79, 132, 133	D
<code>\setenumextmeta</code>	136, 144	<code>\d</code>
<code>\setenumext</code>	32, 133, 135, 136, 139	217
Counters defined by <code>enumext</code> :		<code>\DeclareDocumentEnvironment</code>
<code>enumXiii</code>	30, 42	555
<code>enumXii</code>	30, 42	dim commands:
<code>enumXiv</code>	30, 42	<code>\dim_abs:n</code>
<code>enumXi</code>	30, 42	3505, 3510
<code>enumXviii</code>	30, 42	<code>\dim_add:Nn</code>
<code>enumXvii</code>	30, 42, 124	3147, 4129, 4367, 4398
<code>enumXvi</code>	30, 42	<code>\dim_compare:nNnTF</code>
<code>enumXv</code>	30, 42	1034, 1050, 1063, 1075, 1353,
cs commands:		1365, 1393, 1405, 1432, 1444, 1521, 1529, 1643, 1672,
<code>\cs_generate_variant:Nn</code>	196, 197, 602, 618, 865,	2684, 2692, 3142, 3502, 3507, 3513, 3519, 3521, 3523,
881, 2459, 2464, 2540, 2885, 3531, 4307, 5429		3676, 3723, 3834, 3851, 4104, 4344, 4360, 4375, 4391,
<code>\cs_if_exist:NnTF</code>	572	4504, 4569, 5037
<code>\cs_if_exist_p:N</code>	3461, 5073	<code>\dim_compare:nTF</code>
<code>\cs_new:Nn</code>	211	2652, 3765, 3910
<code>\cs_new:Npn</code>	221, 1820, 1829, 1837, 2421, 2430, 2438,	<code>\dim_eval:n</code>
5278, 5287, 5296		968, 4187, 4274
<code>\cs_new_eq:NN</code>	363, 364, 369, 370, 398, 399, 402, 403	<code>\dim_gset_eq:NN</code>
<code>\cs_new_protected:Nn</code>	227, 235, 261, 292, 322, 328,	4513, 4578
334, 340, 346, 354, 374, 421, 425, 443, 455, 473, 485,		<code>\dim_gzero:N</code>
501, 517, 530, 551, 741, 798, 845, 958, 1109, 1113,		4564, 4624
1117, 1121, 1125, 1129, 1133, 1137, 1141, 1145, 1149,		<code>\dim_new:N</code>
1153, 1157, 1161, 1165, 1169, 1204, 1216, 1249, 1266,		52, 59, 60, 61, 81, 123, 124, 136, 143, 144,
1277, 1294, 1320, 1341, 1466, 1492, 1512, 1545, 1567,		178, 180, 181, 187
1602, 1608, 1714, 1728, 1742, 1753, 1764, 1775, 1786,		<code>\dim_set:Nn</code>
1797, 1878, 1981, 1994, 2011, 2032, 2060, 2065, 2090,		600, 1002, 2686, 2694, 3129, 3133, 3138,
2130, 2140, 2183, 2198, 2205, 2214, 2219, 2224, 2229,		3144, 3231, 3505, 3510, 3512, 3515, 3516, 3520, 3522,
2238, 2243, 2248, 2465, 2489, 2496, 2520, 2527, 2541,		3525, 3526, 3528, 3679, 3726, 3764, 3836, 3853, 3909,
2777, 2796, 2905, 2924, 2955, 2994, 3009, 3037, 3067,		4118, 4223, 4310, 4346, 4353, 4377, 4384, 4439, 4488,
3095, 3108, 3121, 3150, 3163, 3241, 3251, 3262, 3278,		4506, 4571, 4781, 5039
3294, 3420, 3438, 3472, 3484, 3606, 3635, 3664, 3671,		<code>\dim_set_eq:NN</code>
3701, 3718, 3740, 3762, 3776, 3806, 3830, 3847, 3872,		709, 755, 822, 3226, 3543, 3585, 3687,
3886, 3907, 3918, 4086, 4289, 4303, 4308, 4332, 4342,		3861, 4446, 4449, 4450, 4495, 4498, 4499, 4774, 4845,
4373, 4502, 4521, 4567, 4586, 4651, 4678, 4685, 4694,		5120
4704, 4729, 4870, 4915, 4946, 4952, 4973, 5029, 5149		<code>\dim_sub:Nn</code>
<code>\cs_new_protected:Npn</code>	198, 199, 203, 207, 406, 570,	3770, 3915, 4362, 4393
587, 597, 603, 722, 766, 830, 852, 866, 1641, 1670,		<code>\dim_use:N</code>
1846, 1865, 1935, 1968, 2070, 2253, 2375, 2385, 2407,		1035, 1043, 1644, 1654, 2530, 2533, 2538,
2415, 2451, 2460, 2616, 2679, 2704, 2742, 2746, 2839,		2696, 3246, 3248, 3301, 3677, 3681, 3682, 3684, 3724,
2843, 2876, 2935, 2974, 3047, 3088, 3195, 3214, 3343,		3729, 3730, 3736, 3767, 3772
3347, 3361, 3365, 3383, 3387, 3397, 3409, 3454, 3500,		<code>\dim_zero:N</code>
3534, 3574, 3617, 3826, 4095, 4102, 4109, 4214, 4233,		3577, 3690, 3862, 4132
4263, 4404, 4453, 4668, 4735, 4742, 4756, 4764, 4769,		<code>\dim_zero_new:N</code>
4779, 4939, 4979, 4986, 5001, 5010, 5024, 5066, 5171,		569
5184, 5244, 5367, 5379, 5403, 5415, 5453, 5463, 5471,		<code>\c_zero_dim</code>
5493		1037, 1051, 1064, 1076, 1644, 1672, 2654,
<code>\cs_new_protected_nopar:Nn</code>	3975, 4019, 4027,	2684, 2692, 3129, 3142, 3502, 3507, 3513, 3520, 3677,
4035, 4714, 4722, 4853, 4958, 4966, 5132		3724, 3767, 3834, 3851, 3912, 4104, 4344, 4360, 4375,
<code>\cs_new_protected_nopar:Npn</code>	3967, 3983, 4785,	4391, 4504, 4569, 5037
4831, 5085, 5106		<code>\dimeval</code>
<code>\cs_set:Npn</code>	2551, 2588, 5177	2339
<code>\cs_set_eq:NN</code>	3464, 4641, 4642, 4833, 4904, 4905,	
5076, 5108		E
<code>\cs_set_protected:Nn</code>	1032, 1048, 1061, 1073	<code>\end</code>
<code>\cs_set_protected:Npn</code>	32, 39, 56, 64, 79, 85, 127,	2493, 2524, 3708, 3877, 4169, 4324, 5252, 5262, 5270


```

\l__enumext_after_list_args_viii_tl .. 1171,
    5130
\__enumext_after_list_vii: 119, 122, 4649, 4685,
    4685
\__enumext_after_list_viii: ... 128, 4913, 4952,
    4952
\__enumext_after_stop_list: 52, 105, 1109, 1117,
    3756
\__enumext_after_stop_list_v: 1125, 1133, 3904
\l__enumext_after_stop_list_v_tl ..... 1135
\__enumext_after_stop_list_vii: .. 122, 1141,
    1157, 4688
\l__enumext_after_stop_list_vii_tl ... 1159
\__enumext_after_stop_list_viii: . 1161, 4955
\l__enumext_after_stop_list_viii_tl ... 1163
\l__enumext_align_label_pos_v_str 3125, 3490
\l__enumext_align_label_pos_X_str ..... 64
\l__enumext_align_label_vii_str ..... 4820
\l__enumext_align_label_viii_str ..... 5099
\l__enumext_align_label_X_str ..... 170
\c__enumext_all_envs_clist . 191, 650, 907, 1093,
    1108, 1189, 1713
\c__enumext_all_families_seq .. 135, 5335, 5361
\__enumext_anskey_env_file_if_writable:n 86,
    2853, 2853
\__enumext_anskey_env_file_if_-
    writable:nTF ..... 2853, 2878
\__enumext_anskey_env_file_write:nn 86, 2876,
    2885, 2940
\l__enumext_anskey_env_force_eol_bool .. 88,
    2826, 2942
\c__enumext_anskey_env_hidden_space_str 32,
    88, 107, 2946
\l__enumext_anskey_env_overwrite_bool 2834,
    2859
\__enumext_anskey_env_safe_inner: . 87, 2900,
    2905, 2924
\__enumext_anskey_env_safe_inner:n .... 87
\__enumext_anskey_env_safe_outer: . 87, 2888,
    2905, 2905
\__enumext_anskey_env_unknown:n 86, 2837, 2839,
    2839
\__enumext_anskey_env_unknown:nn . 2839, 2841,
    2843
\l__enumext_anskey_level_int .. 16, 2798, 2799
\__enumext_anskey_safe_inner: . 85, 2771, 2777,
    2796
\__enumext_anskey_safe_inner:n ..... 84
\__enumext_anskey_safe_outer: . 84, 2758, 2777,
    2777
\__enumext_anskey_show_wrap_arg:n . 83, 2679,
    2679, 2708, 2723
\__enumext_anskey_show_wrap_left:n 83, 2624,
    2704, 2704
\__enumext_anskey_unknown:n 84, 2726, 2740, 2742
\__enumext_anskey_unknown:nn . 2726, 2744, 2746
\__enumext_anskey_wrapper:n ..... 2336, 2702
\l__enumext_anspic_above_int . 135, 4311, 4312,
    4314
\__enumext_anspic_args:nnn 113, 115, 4195, 4211,
    4289
\l__enumext_anspic_args_seq 113, 115, 135, 4209,
    4323, 4336
\l__enumext_anspic_below_int . 135, 4311, 4312,
    4315
\l__enumext_anspic_body_box ... 135, 4222, 4225
\__enumext_anspic_body_dim:n . 114, 4195, 4214,
    4266
\l__enumext_anspic_body_htdp_dim .. 114, 135,
    4223, 4277
__enumext_anspic_exec: ..... 4195
\__enumext_anspic_exec: ... 113, 116, 4164, 4332
\__enumext_anspic_label:nn 114, 4195, 4233, 4269,
    4284
\l__enumext_anspic_label_above_bool ... 135,
    4047, 4050, 4114, 4179, 4216, 4267, 4294
\l__enumext_anspic_label_box .. 135, 4117, 4120
\l__enumext_anspic_label_htdp_dim . 112, 135,
    4118, 4124, 4189, 4276
\__enumext_anspic_label_pos:nnn .. 114, 4195,
    4263, 4292
\l__enumext_anspic_label_sep_skip 4057, 4126,
    4190, 4279, 4296
\l__enumext_anspic_layout_style_tl 4059, 4334,
    4339
\l__enumext_anspic_mini_pos_str .. 135, 4048,
    4051, 4321
\l__enumext_anspic_mini_width_dim 135, 4235,
    4310, 4321
\__enumext_anspic_print:n 115, 4195, 4303, 4307,
    4336, 4339
\__enumext_anspic_row:n .. 115, 4195, 4305, 4308
\__enumext_anspic_start_list_tag: 3991, 4019,
    4291
\__enumext_anspic_stop_list_tag: . 3991, 4035,
    4301
\__enumext_anspic_stop_start_list_tag: 3991,
    4027, 4293
\__enumext_at_begin_document:n .. 37, 199, 199,
    361, 367
\l__enumext_base_line_fix_bool 49, 134, 954, 963,
    986, 5258, 5263
\__enumext_before_args_exec: 52, 104, 122, 1109,
    1109, 3721
\__enumext_before_args_exec_v: 1125, 1125, 3833
\__enumext_before_args_exec_vii: . 1141, 1141,
    4682
\__enumext_before_args_exec_viii: 1145, 4949
\__enumext_before_env:nn ..... 203, 207
\__enumext_before_keys_exec: .. 52, 1109, 1113,
    3793
\__enumext_before_keys_exec_v: 1125, 1129, 3931
\__enumext_before_keys_exec_vii ..... 1141
\__enumext_before_keys_exec_vii: . 1149, 4635
\__enumext_before_keys_exec_viii: 1153, 4898
\__enumext_before_list: .. 104, 3718, 3718, 3787
\__enumext_before_list_v: ... 3830, 3830, 3926
\__enumext_before_list_vii: ... 122, 4630, 4678,
    4678
\__enumext_before_list_viii: .. 128, 4894, 4946,
    4946
\l__enumext_before_no_starred_key_v_tl 1131
\l__enumext_before_no_starred_key_vii_-
    tl ..... 1151
\l__enumext_before_no_starred_key_viii_-
    tl ..... 1155
\l__enumext_before_starred_key_v_tl ... 1127
\l__enumext_before_starred_key_vii_tl . 1143

```

```

\l__enumext_before_starred_key_viii_tl 1147
\__enumext_calc_hspace:NNNNNNN 100, 3500, 3500,
    3531, 3536, 3578
\__enumext_check_ans_active: 72, 104, 122, 2130,
    2130, 3722, 4681
\g__enumext_check_ans_item_tl . . . . . 90
\g__enumext_check_ans_key_bool 73, 74, 145, 336,
    2189, 2195, 2965
\l__enumext_check_ans_key_bool 73, 2115, 2120,
    2186, 2192
\__enumext_check_ans_key_hook: . . 73, 105, 122,
    2183, 2183, 3757, 4689
\__enumext_check_ans_level: 72, 2130, 2136, 2140
\__enumext_check_ans_log: 74, 88, 2229, 2229, 2969
\__enumext_check_ans_log_msg_greater: 2229,
    2235, 2248
\__enumext_check_ans_log_msg_less: 2229, 2233,
    2238
\__enumext_check_ans_log_msg_same_ok: 2229,
    2234, 2243
\__enumext_check_ans_msg_greater: 2205, 2211,
    2224
\__enumext_check_ans_msg_less: 2205, 2209, 2214
\__enumext_check_ans_msg_same_ok: 2205, 2210,
    2219
\__enumext_check_ans_show: 73, 74, 88, 2205, 2205,
    2967
\l__enumext_check_answers_bool 71, 72, 84, 87, 93,
    145, 2093, 2119, 2134, 2467, 2491, 2498, 2522, 2760,
    2889, 3083, 3199, 3233, 4800
\__enumext_check_starred_cmd:n 36, 75, 90, 127,
    2253, 2253, 3938, 4177, 4912
\g__enumext_check_starred_cmd_int . . 97, 145,
    2256, 2262, 2267, 3418, 4245, 5036
\l__enumext_check_start_line_env_tl . 36, 145,
    299, 307, 315, 2259, 2265, 2268
\l__enumext_columns_sep_v_dim 3851, 3853, 3861
\l__enumext_columns_sep_vii_dim . . 4344, 4346,
    4355, 4367, 4443, 4867
\l__enumext_columns_sep_viii_dim . 4375, 4377,
    4386, 4398, 4492, 5146
\l__enumext_columns_v_int 1486, 1504, 1675, 3849,
    3857, 3869, 3874
\l__enumext_columns_vii_int . . 4349, 4352, 4356,
    4365, 4407, 4411, 4414, 4420, 4426, 4430, 4861, 4875
\l__enumext_columns_viii_int . 4380, 4383, 4387,
    4396, 4456, 4460, 4463, 4469, 4475, 4479, 5140, 5155
\l__enumext_counter_i_tl . . . . . 32, 579
\l__enumext_counter_ii_tl . . . . . 32, 580
\l__enumext_counter_iii_tl . . . . . 32, 581
\l__enumext_counter_iv_tl . . . . . 32, 582
\g__enumext_counter_styles_tl . 30, 42, 52, 590,
    608
\l__enumext_counter_v_tl . . . . . 32, 583
\l__enumext_counter_vi_tl . . . . . 32, 584
\l__enumext_counter_vii_tl . . . . . 32, 585
\l__enumext_counter_viii_tl . . . . . 32, 586
\l__enumext_current_widest_dim 30, 52, 614, 710,
    756, 823
\__enumext_def_meta_key:nnn . . . 136, 5381, 5409,
    5415, 5429
\__enumext_default_item:n . . . 3195, 3195, 3259
\__enumext_define_counter:Nn . 30, 570, 570, 579,
    580, 581, 582, 583, 584, 585, 586
\__enumext_endminipage: . 38, 361, 370, 564, 4558,
    4855, 5134
\g__enumext_envir_name_tl 35, 22, 271, 285, 344,
    2063, 2068, 2078, 2217, 2222, 2227, 2241, 2246, 2251
\l__enumext_envir_name_tl 35, 36, 96, 22, 241, 251,
    298, 306, 314, 3337, 4082, 5793, 5796, 5803, 5806,
    5813, 5816, 5823, 5826, 5832, 5836, 5842, 5846, 5903,
    5907
\__enumext_execute_after_env: 36, 37, 70, 74, 88,
    2955, 2955, 3804, 4888
\__enumext_fake_item_indent: . 1032, 1032, 3562
\l__enumext_fake_item_indent_v_dim 1051, 1056
\l__enumext_fake_item_indent_v_tl 1053, 3402,
    3406, 3413
\__enumext_fake_item_indent_vii: . 1032, 1061,
    3594
\l__enumext_fake_item_indent_vii_dim . 1064,
    1068
\l__enumext_fake_item_indent_vii_tl . . 1066,
    4850
\__enumext_fake_item_indent_viii: 1032, 1073,
    3599
\l__enumext_fake_item_indent_viii_dim 1076,
    1080
\l__enumext_fake_item_indent_viii_tl . 1078,
    5125
\l__enumext_fake_item_indent_X_tl . . . . . 85
\__enumext_fake_make_label_vii:n . 124, 4785,
    4785, 4842
\__enumext_fake_make_label_viii:n 5066, 5085,
    5117
\__enumext_filter_first_level:n . . 134, 5278,
    5278, 5312, 5323
\__enumext_filter_first_level_key:n 134, 5278,
    5283, 5287
\__enumext_filter_first_level_pair:nn . 135,
    5278, 5284, 5296
\__enumext_filter_save_key:n . . 78, 2382, 2390,
    2413, 2419, 2421, 2421, 5197, 5203, 5209, 5215, 5221,
    5227
\__enumext_filter_save_key_key:n . . 78, 2421,
    2426, 2430
\__enumext_filter_save_key_pair:nn 78, 2421,
    2427, 2438
\__enumext_filter_series:n 66, 1820, 1820, 1858,
    1870, 1875
\__enumext_filter_series_key:n 66, 1820, 1825,
    1829
\__enumext_filter_series_pair:nn . . 66, 1820,
    1826, 1837
\__enumext_first_item_tmp_vii: 121, 123, 4641,
    4714, 4714
\__enumext_first_item_tmp_viii: . . 128, 4904,
    4958, 4958
\g__enumext_footnote_standar_arg_seq . . 164,
    438, 449, 452
\g__enumext_footnote_standar_int 164, 432, 435,
    437, 440
\g__enumext_footnote_standar_int_seq . . 164,
    440, 445, 448, 453
\g__enumext_footnote_starred_arg_seq . . 164,
    468, 479, 482
\g__enumext_footnote_starred_int 164, 462, 465,
    467, 470
\g__enumext_footnote_starred_int_seq . . 164,

```

470, 475, 478, 483
 __enumext_footnotes_key_bool 38
 \l__enumext_footnotes_key_bool 33, 38, 154, 382, 386, 393, 494, 510, 524, 537
 __enumext_footnotetext:nn . . 421, 421, 450, 480
 __enumext_foreach_add_body:n . 138, 5430, 5490, 5493
 \l__enumext_foreach_after_tl 5434, 5502
 \l__enumext_foreach_before_tl 5432, 5497
 \g__enumext_foreach_default_keys_tl . . . 137
 \l__enumext_foreach_default_keys_tl . . . 114, 5452, 5473
 __enumext_foreach_keyans:nn . . 138, 5430, 5469, 5471
 \l__enumext_foreach_name_prop_tl . . 114, 5475, 5500
 \l__enumext_foreach_print_seq 114, 5485, 5491, 5495
 \l__enumext_foreach_sep_tl 5444, 5491
 \l__enumext_foreach_start_int 5436, 5487
 \l__enumext_foreach_step_int 5440, 5488
 \l__enumext_foreach_stop_int . 5438, 5480, 5482, 5489
 __enumext_foreach_wrapper:n 5442, 5498
 __enumext_getkeyans:nn . . 132, 5166, 5180, 5184
 __enumext_getkeyans_aux:n 132, 5166, 5168, 5171
 \l__enumext_hyperref_bool 33, 38, 154, 379, 396, 413, 2669, 3071, 4794
 __enumext_hypertarget:nn 38, 372, 398, 402, 418
 __enumext_if_is_int:n 215
 __enumext_if_is_int:nTF 215, 854, 868
 __enumext_internal_mini_page: 41, 102, 121, 551, 551, 3609, 4654
 __enumext_is_not_nested: . 30, 35, 102, 121, 235, 235, 3608, 4653
 __enumext_is_on_first_level: . 30, 35, 102, 121, 235, 261, 3615, 4666
 \g__enumext_item_anskey_int 84, 90, 145, 331, 358, 359, 2202, 2618, 3085
 __enumext_item_answer_diff: . . 73, 74, 88, 2198, 2198, 2962
 \g__enumext_item_answer_diff_int . 73, 74, 145, 332, 2200, 2207, 2231
 \l__enumext_item_column_pos_vii_int 123, 4414, 4420, 4426, 4430, 4437, 4725, 4861, 4864
 \l__enumext_item_column_pos_viii_int . . 128, 4463, 4469, 4475, 4479, 4486, 4969, 5140, 5143
 \l__enumext_item_column_pos_X_int 170
 \g__enumext_item_count_all_vii_int 123, 4438, 4726, 4875, 4883
 \g__enumext_item_count_all_viii_int 128, 4487, 4970, 5154, 5163
 \g__enumext_item_count_all_X_int 170
 \g__enumext_item_number_bool 145
 \l__enumext_item_number_bool 72, 152, 2152, 2157, 2161, 2165, 2178, 2803, 2926, 3202, 3236, 4803
 \g__enumext_item_number_int 72, 73, 145, 330, 357, 359, 2151, 2156, 2160, 2164, 2177, 2202, 3201, 3235, 4802
 __enumext_item_peek_args_vii: 123, 4722, 4727, 4729
 __enumext_item_peek_args_viii: . . 128, 4966, 4971, 4973
 __enumext_item_starred_exec: . 94, 3214, 3241, 3283, 3304
 __enumext_item_starred_exec:nn . . 3214, 3214, 3257
 \l__enumext_item_starred_vii_bool 4744, 4758, 4807
 \l__enumext_item_starred_viii_bool 4988, 5003, 5095, 5126
 \l__enumext_item_starred_X_bool 170
 __enumext_item_std:w . 37, 93, 97, 361, 365, 3205, 3211, 3239, 3402, 3406, 3413
 \g__enumext_item_symbol_aux_tl . 93, 118, 3219, 3222, 3247, 3291, 3311
 \g__enumext_item_symbol_aux_vii_tl 4766, 4809, 4812, 4816, 4818
 \g__enumext_item_symbol_aux_X_tl 170
 \l__enumext_item_symbol_sep_vii_dim . . 4774, 4781, 4815, 4817
 \l__enumext_item_symbol_vii_tl 4812
 \l__enumext_item_text_vii_box 4834, 4859
 \l__enumext_item_text_viii_box . . . 5109, 5138
 \l__enumext_item_text_X_box 170
 \l__enumext_item_width_vii_dim . . 4353, 4362, 4441, 4449, 4450
 \l__enumext_item_width_viii_dim . . 4384, 4393, 4490, 4498, 4499
 \l__enumext_item_width_X_dim 170
 \l__enumext_item_wrap_key_bool . 98, 145, 3155, 3175, 3426, 3433, 3460, 4239, 4257, 4989, 5004, 5072
 \l__enumext_itemindent_X_dim 56
 \l__enumext_itemsep_i_skip . . . 1347, 1354, 1357, 1359, 1366, 1370, 1373, 1375, 1515, 1522, 1524, 1525, 1530, 1534, 1536, 1537
 \l__enumext_itemsep_ii_skip . . 1387, 1394, 1397, 1399, 1406, 1410, 1413, 1415
 \l__enumext_itemsep_iii_skip . 1426, 1433, 1436, 1438, 1445, 1449, 1452, 1454
 \l__enumext_itemsep_vii_skip 4881
 \l__enumext_itemsep_viii_skip 5161
 \l__enumext_joined_item_aux_vii_int . . 4435, 4436, 4437, 4438, 4444
 \l__enumext_joined_item_aux_viii_int . 4484, 4485, 4486, 4487, 4493
 \l__enumext_joined_item_aux_X_int 170
 __enumext_joined_item_vii:w . . 123, 4722, 4732, 4733, 4735
 \l__enumext_joined_item_vii_int . . 4406, 4407, 4410, 4412, 4418, 4423, 4428, 4433, 4435, 4441
 __enumext_joined_item_viii:w . 128, 129, 4966, 4976, 4977, 4979
 \l__enumext_joined_item_viii_int . 4455, 4456, 4459, 4461, 4467, 4472, 4477, 4482, 4484, 4490
 \l__enumext_joined_item_X_int 170
 \l__enumext_joined_width_vii_dim . 4439, 4446, 4449, 4836, 4844
 \l__enumext_joined_width_viii_dim 4488, 4495, 4498, 5111, 5119
 \l__enumext_joined_width_X_dim 170
 __enumext_keyans_addto_prop:n 89, 2974, 2974, 3415, 4242
 __enumext_keyans_addto_seq:n . 90, 3047, 3047, 3417, 4244
 __enumext_keyans_addto_seq_link: 3047, 3065, 3067, 5035
 __enumext_keyans_default_item:n . . 97, 3397, 3397, 3434

`\l__enumext_keyans_env_bool` [22](#), [3640](#), [3653](#), [3813](#), [3903](#)
`__enumext_keyans_fake_item_indent`: [..](#) [1032](#), [1048](#), [3552](#)
`\l__enumext_keyans_level_h_int` [..](#) [127](#), [16](#), [783](#), [807](#), [2787](#), [2915](#), [3025](#), [4660](#), [4921](#), [4922](#)
`\l__enumext_keyans_level_int` [..](#) [16](#), [1635](#), [2783](#), [2911](#), [3020](#), [3165](#), [3812](#), [3817](#), [4205](#)
`__enumext_keyans_make_label`: [.](#) [98](#), [3438](#), [3438](#), [3550](#)
`__enumext_keyans_make_label_box`: [3438](#), [3442](#), [3447](#), [3484](#)
`__enumext_keyans_make_label_std`: [3438](#), [3450](#), [3472](#)
`__enumext_keyans_mini_right_cmd:n` [63](#), [1637](#), [1670](#), [1670](#)
`__enumext_keyans_mini_set_vskip`: [.....](#) [59](#)
`__enumext_keyans_minipage_add_space`: [1466](#), [1492](#), [3842](#)
`__enumext_keyans_minipage_set_skip`: [.](#) [1466](#), [1466](#), [1494](#)
`__enumext_keyans_multi_addvspace`: [1266](#), [1277](#), [3866](#)
`__enumext_keyans_multi_set_vskip`: [56](#), [1266](#), [1266](#), [1279](#)
`__enumext_keyans_multicols_start`: [3830](#), [3845](#), [3847](#)
`__enumext_keyans_multicols_stop`: [1674](#), [3830](#), [3872](#), [3901](#)
`__enumext_keyans_name_and_start`: [30](#), [36](#), [127](#), [292](#), [292](#), [3814](#), [4093](#), [4926](#)
`__enumext_keyans_parse_keys:n` [3826](#), [3826](#), [3925](#)
`__enumext_keyans_pic_arg_two`: [112](#), [4086](#), [4109](#), [4140](#)
`\l__enumext_keyans_pic_level_int` [..](#) [16](#), [1616](#), [2791](#), [2919](#), [2977](#), [3015](#), [3050](#), [4088](#), [4089](#)
`__enumext_keyans_pic_parse_keys:n` [4086](#), [4095](#), [4139](#)
`__enumext_keyans_pic_safe_exec`: [.](#) [111](#), [4086](#), [4086](#), [4138](#)
`__enumext_keyans_pic_skip_abs:N` [.](#) [111](#), [4086](#), [4102](#), [4113](#)
`__enumext_keyans_pos_mark_set`: [91](#), [3121](#), [3121](#), [3158](#), [3190](#)
`__enumext_keyans_pre_itemsep_skip`: [..](#) [1466](#), [1485](#), [1512](#)
`__enumext_keyans_redefine_item`: [..](#) [98](#), [3420](#), [3420](#), [3549](#)
`__enumext_keyans_ref`: [.....](#) [47](#), [830](#), [845](#), [3551](#)
`__enumext_keyans_ref:n` [.....](#) [47](#), [827](#), [830](#), [830](#)
`__enumext_keyans_safe_exec`: [.](#) [3806](#), [3806](#), [3924](#)
`__enumext_keyans_save_item_opt:n` [91](#), [97](#), [3088](#), [3088](#), [3411](#), [4241](#)
`__enumext_keyans_set_item_width`: [108](#), [3907](#), [3907](#), [3934](#)
`__enumext_keyans_show_ans`: [92](#), [3121](#), [3150](#), [3477](#), [3492](#), [4246](#)
`__enumext_keyans_show_item_opt`: [91](#), [97](#), [3088](#), [3095](#), [3414](#), [4254](#)
`__enumext_keyans_show_item_opt_viii`: [..](#) [91](#), [3088](#), [3108](#), [5128](#)
`__enumext_keyans_show_pos`: [92](#), [3121](#), [3163](#), [3478](#), [3493](#), [4247](#)
`__enumext_keyans_starred_item:n` [..](#) [97](#), [3409](#), [3409](#), [3429](#)
`__enumext_keyans_starred_item_star`: [..](#) [130](#), [5001](#), [5029](#), [5097](#)
`__enumext_keyans_start_counter`: [.](#) [3918](#), [3918](#), [3933](#)
`__enumext_keyans_store_ref`: [..](#) [89](#), [2994](#), [2994](#), [3416](#), [4243](#), [5033](#)
`__enumext_keyans_store_ref_aux_i`: [89](#), [2994](#), [3006](#), [3009](#)
`__enumext_keyans_store_ref_aux_ii`: [90](#), [2994](#), [3035](#), [3037](#)
`__enumext_keyans_unknown_keys:n` [.](#) [3332](#), [3338](#), [3343](#), [4083](#)
`__enumext_keyans_unknown_keys:nn` [3332](#), [3345](#), [3347](#)
`__enumext_keyans_wrapper_label:n` [.....](#) [98](#)
`__enumext_keyans_wrapper_label_viii:n` [5066](#), [5066](#), [5102](#)
`__enumext_keyans_wrapper_item_v:n` [3461](#), [3464](#)
`__enumext_keyans_wrapper_item_viii:n` [5073](#), [5077](#)
`__enumext_keyans_wrapper_label:n` [3438](#), [3454](#), [3480](#), [3495](#), [4251](#)
`__enumext_keyans_wrapper_opt_v:n` [....](#) [3103](#)
`__enumext_keyans_wrapper_opt_viii:n` [..](#) [3116](#)
`\l__enumext_label_copy_i_tl` [..](#) [2584](#), [3013](#), [3018](#), [3023](#), [3028](#)
`\l__enumext_label_copy_v_tl` [.....](#) [3023](#)
`\l__enumext_label_copy_vi_tl` [.....](#) [3018](#)
`\l__enumext_label_copy_vii_tl` [2560](#), [2571](#), [2600](#), [3013](#)
`\l__enumext_label_copy_viii_tl` [.....](#) [3028](#)
`\l__enumext_label_copy_X_tl` [.....](#) [156](#)
`\l__enumext_label_fill_left_v_tl` [.....](#) [3476](#)
`\l__enumext_label_fill_left_X_tl` [.....](#) [85](#)
`\l__enumext_label_fill_right_v_tl` [....](#) [3481](#)
`\l__enumext_label_fill_right_X_tl` [.....](#) [85](#)
`\l__enumext_label_font_style_v_tl` [3479](#), [3494](#), [4250](#), [4258](#)
`\l__enumext_label_font_style_vii_tl` [...](#) [4822](#)
`\l__enumext_label_font_style_viii_tl` [..](#) [5101](#)
`\l__enumext_label_i_tl` [.....](#) [702](#)
`\l__enumext_label_ii_tl` [.....](#) [702](#)
`\l__enumext_label_iii_tl` [.....](#) [702](#)
`\l__enumext_label_iv_tl` [.....](#) [702](#)
`__enumext_label_style:Nnn` [30](#), [42](#), [603](#), [603](#), [618](#), [707](#), [753](#), [818](#), [820](#)
`\l__enumext_label_v_tl` [90](#), [815](#), [2982](#), [3055](#), [3124](#), [3928](#), [4117](#)
`\l__enumext_label_vi_tl` [90](#), [815](#), [2979](#), [3052](#), [4251](#), [4259](#)
`\l__enumext_label_vii_tl` [.](#) [748](#), [4753](#), [4776](#), [4783](#)
`\l__enumext_label_viii_tl` [748](#), [4998](#), [5027](#), [5031](#)
`\l__enumext_label_width_by_box` [..](#) [52](#), [599](#), [600](#)
`__enumext_label_width_by_box:Nn` [42](#), [597](#), [597](#), [602](#), [614](#), [878](#), [3123](#)
`\l__enumext_labelsep_v_dim` [...](#) [3144](#), [3856](#), [4129](#), [4253](#)
`\l__enumext_labelsep_vii_dim` [.](#) [2686](#), [4348](#), [4358](#), [4442](#), [4718](#), [4774](#), [4829](#), [4838](#)
`\l__enumext_labelsep_viii_dim` [4379](#), [4389](#), [4491](#), [4962](#), [5039](#), [5104](#), [5113](#)
`\l__enumext_labelwidth_v_dim` [.](#) [823](#), [3134](#), [3139](#), [3160](#), [3192](#), [3490](#), [3856](#), [4129](#), [4248](#)
`\l__enumext_labelwidth_vii_dim` [...](#) [2689](#), [4348](#),

4357, 4442, 4718, 4820, 4837
 \l__enumext_labelwidth_viii_dim . 4379, 4388,
 4491, 4962, 5046, 5063, 5099, 5112
 \l__enumext_leftmargin_tmp_v_bool . 112, 4111
 \l__enumext_leftmargin_tmp_X_bool 56
 \l__enumext_leftmargin_tmp_X_dim 56
 \l__enumext_leftmargin_X_dim 56
 __enumext_level: 211, 211, 732, 734, 743, 745, 1035,
 1039, 1043, 1111, 1115, 1119, 1123, 1206, 1208, 1210,
 1212, 1254, 1256, 1258, 1260, 1264, 1298, 1304, 1309,
 1311, 1314, 1317, 1330, 1333, 1644, 1648, 1654, 1717,
 1719, 1721, 1724, 1731, 1733, 1735, 1738, 2377, 2379,
 2381, 2409, 2410, 2412, 2469, 2477, 2481, 2485, 2696,
 2700, 3204, 3205, 3209, 3210, 3211, 3219, 3227, 3228,
 3231, 3238, 3239, 3243, 3246, 3248, 3282, 3284, 3285,
 3287, 3290, 3301, 3302, 3305, 3306, 3308, 3646, 3659,
 3666, 3674, 3677, 3679, 3681, 3682, 3683, 3684, 3687,
 3692, 3698, 3704, 3711, 3724, 3726, 3729, 3730, 3732,
 3736, 3742, 3767, 3772, 3778, 3780, 3790, 3792
 \l__enumext_level_h_int 121, 16, 244, 267, 280, 769,
 800, 1623, 2148, 2168, 2579, 3654, 4655, 4656
 \l__enumext_level_int . 102, 16, 213, 254, 266, 281,
 553, 1218, 1343, 1622, 2142, 2174, 2556, 2566, 2572,
 2578, 2585, 2594, 2599, 2957, 3566, 3610, 3611, 3622,
 3630, 3644, 3657, 3688, 3821, 4201, 4698, 4708, 4934,
 5833, 5837, 5843, 5847
 __enumext_list_arg_two_i: 3532
 __enumext_list_arg_two_ii: 3532
 __enumext_list_arg_two_iii: 3532
 __enumext_list_arg_two_iv: 3532
 __enumext_list_arg_two_v: . 98, 3532, 3930, 4112
 __enumext_list_arg_two_vii: 3572, 4634
 __enumext_list_arg_two_viii: 3572, 4897
 \l__enumext_listoffset_v_dim . 3858, 3912, 3915
 \l__enumext_listparindent_vii_dim 4845, 4849
 \l__enumext_listparindent_viii_dim 5120, 5124
 __enumext_log_answer_vars: . 37, 346, 354, 2964
 __enumext_log_global_vars: . 37, 346, 346, 2963
 __enumext_make_label: 94, 3262, 3262, 3560
 __enumext_make_label_box: . . . 3262, 3266, 3271,
 3294
 __enumext_make_label_std: . . . 3262, 3274, 3278
 \l__enumext_mark_answer_sym_tl 80, 2321, 2535,
 2712, 3146, 5043, 5050
 \l__enumext_mark_answer_sym_v_tl . 3146, 3178
 \l__enumext_mark_answer_sym_viii_tl . . . 5043
 \l__enumext_mark_position_str 118, 2327, 2328,
 2329, 2533, 3148, 5044, 5061
 \l__enumext_mark_position_v_str . . 118, 3148
 \l__enumext_mark_position_viii_str 118, 5044,
 5061
 \l__enumext_mark_ref_sym_tl . . 2309, 2674, 3079
 \l__enumext_mark_sep_tmpa_dim 118, 3124, 3134,
 3139
 \l__enumext_mark_sep_tmpb_dim 118, 3129, 3133,
 3138, 3147
 \l__enumext_mark_sym_sep_dim . 2324, 2684, 2686,
 2689, 2692, 2694
 \l__enumext_mark_sym_sep_v_dim . . . 3142, 3144,
 3147, 3160, 3192
 \l__enumext_mark_sym_sep_viii_dim 5037, 5039,
 5046, 5063
 \l__enumext_meta_path_tl . 114, 5405, 5406, 5408,
 5409
 \c__enumext_meta_paths_prop 136, 5381
 __enumext_mini_addvspace_vii: 61, 1602, 1602,
 4516
 __enumext_mini_addvspace_viii: 61, 1602, 1608,
 4581
 __enumext_mini_env* 551
 __enumext_mini_page 1654, 1681, 3736, 3843, 4518,
 4583, 4604
 __enumext_mini_right_cmd:n . 62, 63, 1639, 1641,
 1641
 __enumext_mini_set_vskip_vii: 60, 1545, 1545,
 1604
 __enumext_mini_set_vskip_viii: 60, 1545, 1567,
 1610
 __enumext_minipage:w 38, 361, 369, 558, 4541, 4844,
 5119
 \l__enumext_minipage_active_v_bool 3840, 3863,
 3888
 \g__enumext_minipage_active_vii_bool . . 119,
 4530, 4539, 4561
 \l__enumext_minipage_active_vii_bool . 4512,
 4523
 \g__enumext_minipage_active_viii_bool 4594,
 4602, 4621
 \l__enumext_minipage_active_viii_bool 4577,
 4588
 \g__enumext_minipage_active_X_bool . . . 170
 \l__enumext_minipage_active_X_bool 72
 __enumext_minipage_add_space: . 57, 104, 1294,
 1320, 3734
 \g__enumext_minipage_after_skip 72, 1549, 1561,
 4559, 4619
 \l__enumext_minipage_after_skip . . 56, 104, 72,
 1307, 1347, 1349, 1354, 1357, 1361, 1366, 1370, 1373,
 1377, 1389, 1394, 1397, 1401, 1406, 1410, 1413, 1417,
 1428, 1433, 1436, 1440, 1445, 1449, 1452, 1456, 1468,
 1482, 1515, 1517, 1522, 1524, 1526, 1530, 1534, 1536,
 1538, 1569, 1582, 1596, 1650, 1677, 3898
 \g__enumext_minipage_center_vii_bool . 4545,
 4562
 \g__enumext_minipage_center_viii_bool 4606,
 4622
 \g__enumext_minipage_center_X_bool 170
 \l__enumext_minipage_hsep_v_dim 3838
 \l__enumext_minipage_hsep_vii_dim 4510
 \l__enumext_minipage_hsep_viii_dim . . . 4575
 \l__enumext_minipage_left_skip 72, 1469, 1547,
 1552, 1556, 1570, 1574, 1588, 1606, 1612
 \l__enumext_minipage_left_v_dim . . 3836, 3843
 \l__enumext_minipage_left_vii_dim 4506, 4518
 \l__enumext_minipage_left_viii_dim 4571, 4583
 \l__enumext_minipage_left_X_dim 72
 \g__enumext_minipage_right_skip 72, 1548, 1553,
 1557, 4544, 4605
 \l__enumext_minipage_right_skip . 56, 72, 1296,
 1302, 1307, 1309, 1311, 1470, 1471, 1477, 1482, 1483,
 1484, 1489, 1571, 1578, 1592, 1656, 1683
 \l__enumext_minipage_right_v_dim . 1672, 1681,
 3834, 3838
 \g__enumext_minipage_right_vii_dim 119, 4514,
 4541, 4564
 \l__enumext_minipage_right_vii_dim 119, 4504,
 4509, 4515
 \g__enumext_minipage_right_viii_dim . . 4579,
 4604, 4624


```

\l__enumext_minipage_right_viii_dim .. 4569,
    4574, 4580
\g__enumext_minipage_right_X_dim .. 170
\g__enumext_minipage_right_X_skip .. 170
\__enumext_minipage_set_skip: . 56, 1294, 1294,
    1322
\g__enumext_minipage_stat_int .. 104, 72, 1661,
    1688, 3733, 3744, 3749, 3841, 3890, 3895
\l__enumext_minipage_temp_skip 72, 1368, 1378,
    1381, 1408, 1418, 1421, 1447, 1457, 1460, 1532, 1539,
    1541
\l__enumext_miniright_code_vii_box 4552, 4556
\g__enumext_miniright_code_vii_tl 119, 4547,
    4554, 4563
\l__enumext_miniright_code_viii_box .. 4613,
    4617
\g__enumext_miniright_code_viii_tl 4608, 4615,
    4623
\l__enumext_miniright_code_X_box .. 170
\l__enumext_mode_box_bool .. 623, 3269, 3445
\__enumext_multi_addvspace: 55, 103, 1249, 1249,
    3695
\__enumext_multi_set_vskip: 54, 1204, 1204, 1251
\l__enumext_multicols_above_ii_skip ... 1223
\l__enumext_multicols_above_iii_skip .. 1232
\l__enumext_multicols_above_iv_skip ... 1241
\l__enumext_multicols_above_v_skip 1268, 1282,
    1292, 1483
\l__enumext_multicols_above_X_skip .... 64
\l__enumext_multicols_below_ii_skip .. 1350,
    1359, 1363, 1375, 1380
\l__enumext_multicols_below_iii_skip . 1390,
    1399, 1403, 1415, 1420
\l__enumext_multicols_below_iv_skip .. 1429,
    1438, 1442, 1454, 1459
\l__enumext_multicols_below_v_skip 1272, 1286,
    1484, 1518, 1525, 1527, 1537, 1540, 3880
\l__enumext_multicols_below_X_skip .... 64
\g__enumext_multicols_right_X_skip .... 64
\__enumext_multicols_start: 103, 104, 3671, 3671,
    3738
\__enumext_multicols_stop: 104, 1646, 3701, 3701,
    3754
\__enumext_nested_base_line_fix: 49, 102, 952,
    958, 3626
\__enumext_newlabel:nn 33, 38, 81, 406, 406, 2610,
    3041
\l__enumext_newlabel_arg_one_tl 33, 38, 81, 89,
    156, 2603, 2611, 2673, 3030, 3042, 3077
\l__enumext_newlabel_arg_two_tl 33, 38, 80, 156,
    2559, 2569, 2582, 2597, 2612, 3017, 3022, 3027, 3043
\__enumext_parse_foreach_keys:n .. 5430, 5446,
    5463
\__enumext_parse_foreach_keys:nn . 5430, 5453,
    5465
\__enumext_parse_keys:n 49, 66, 3617, 3617, 3786
\__enumext_parse_keys_vii:n 66, 4629, 4668, 4668
\__enumext_parse_keys_viii:n . 4893, 4939, 4939
\__enumext_parse_save_key:n 77, 2402, 2407, 2407
\__enumext_parse_save_key_vii:n 77, 2397, 2407,
    2415
\__enumext_parse_series:n .. 66, 102, 122, 1846,
    1846, 3625, 4674
\__enumext_parse_store_keys:n ..... 102
\l__enumext_parsep_i_skip ..... 1221, 1225
\l__enumext_parsep_ii_skip ..... 1230, 1234
\l__enumext_parsep_iii_skip ..... 1239, 1243
\l__enumext_parsep_vii_skip ..... 4846
\l__enumext_parsep_viii_skip ..... 5121
\l__enumext_partopsep_v_skip . 1284, 1288, 1479,
    1502
\l__enumext_partopsep_viii_skip ..... 1580
\__enumext_phantomsection: 38, 372, 399, 403, 419
\__enumext_pre_itemsep_skip: .. 57, 1312, 1341,
    1341
\__enumext_print_footnote: .. 421, 443, 507, 512
\__enumext_print_footnote_mini: 421, 473, 534,
    539
\__enumext_print_footnote_standar: 485, 501,
    565
\__enumext_print_footnote_starred: 485, 530,
    545, 549
\__enumext_print_keyans_box:NN 80, 2527, 2527,
    2540, 2688, 2699, 3159, 3191, 5045, 5062
\l__enumext_print_keyans_i_tl .... 5204, 5236
\l__enumext_print_keyans_ii_tl ... 5210, 5237
\l__enumext_print_keyans_iii_tl .. 5216, 5238
\l__enumext_print_keyans_iv_tl ... 5222, 5239
\l__enumext_print_keyans_star_bool 49, 50, 134,
    118, 964, 976, 5259, 5264
\l__enumext_print_keyans_starred_tl 132, 134,
    118, 5198, 5257
\l__enumext_print_keyans_vii_tl 132, 5228, 5240
\l__enumext_print_keyans_X_tl ..... 118
\__enumext_printkeyans:nnn 133, 5233, 5241, 5244
\__enumext_redefine_item: . 94, 3251, 3251, 3559
\l__enumext_ref_key_arg_t ..... 45
\l__enumext_ref_key_arg_tl 37, 724, 725, 737, 768,
    771, 779, 785, 793, 832, 833, 841
\l__enumext_ref_the_count_tl . 45, 37, 730, 736,
    776, 779, 790, 793, 838, 841
\__enumext_register_default_label_wd:Nn 587,
    587, 592, 593, 594, 595, 596
\__enumext_remove_extra_parsep_vii: .. 4648,
    4870, 4870
\__enumext_remove_extra_parsep_viii: . 4911,
    5149, 5149
\l__enumext_renew_counter_v_tl . 839, 847, 849
\l__enumext_renew_counter_vii_tl 777, 802, 804
\l__enumext_renew_counter_viii_tl . 791, 809,
    811
\l__enumext_renew_counter_X_tl ..... 37
\__enumext_renew_footnote: .. 421, 425, 491, 496
\__enumext_renew_footnote_mini: 421, 455, 521,
    526
\__enumext_renew_footnote_standar: 485, 485,
    557
\__enumext_renew_footnote_starred: 485, 517,
    4840, 5115
\__enumext_reset_global_bool: .. 322, 325, 334
\__enumext_reset_global_int: ... 322, 324, 328
\__enumext_reset_global_tl: .... 322, 326, 340
\__enumext_reset_global_vars: . 36, 88, 322, 322,
    2971
\l__enumext_resume_active_bool 66, 68, 46, 1850,
    1970
\__enumext_resume_counter: .. 68, 69, 1968, 1974,
    1981
\__enumext_resume_counter:n . 66, 68, 1939, 1944,

```

[1968](#), [1968](#), [2038](#), [2046](#)
`__enumext_resume_counter_save_ans:` [69](#), [1968](#),
[1979](#), [2011](#)
`__enumext_resume_counter_series:` [69](#), [1968](#),
[1977](#), [1994](#)
`\g__enumext_resume_int` [46](#), [1891](#), [1985](#), [1986](#)
`__enumext_resume_last:n` [66](#), [67](#), [1846](#), [1852](#), [1865](#)
`\l__enumext_resume_name_tl` [46](#), [1887](#), [1895](#), [1898](#),
[1914](#), [1922](#), [1925](#), [1971](#), [1972](#), [2000](#), [2007](#)
`__enumext_resume_save_counter:` [67](#), [105](#), [122](#),
[1878](#), [1878](#), [3760](#), [4692](#)
`__enumext_resume_series:n` [68](#), [1814](#), [1935](#), [1935](#)
`__enumext_resume_starred:` [70](#), [1815](#), [2032](#), [2032](#)
`\g__enumext_resume_vii_int` [46](#), [1918](#), [1990](#), [1991](#)
`\l__enumext_rightmargin_vii_dim` [4360](#), [4364](#),
[4369](#)
`\l__enumext_rightmargin_viii_dim` [4391](#), [4395](#),
[4400](#)
`__enumext_safe_exec:` [41](#), [102](#), [3606](#), [3606](#), [3785](#)
`__enumext_safe_exec_vii:` [41](#), [4628](#), [4651](#), [4651](#)
`__enumext_safe_exec_viii:` [127](#), [4892](#), [4915](#), [4915](#)
`__enumext_scan_tokens:n` [88](#), [198](#), [198](#), [2952](#)
`__enumext_second_part:` [104](#), [3740](#), [3740](#), [3800](#)
`__enumext_second_part_v:` [3830](#), [3886](#), [3939](#)
`\l__enumext_series_name_tl` [68](#), [69](#)
`\l__enumext_series_str` [67](#), [102](#), [122](#), [1812](#), [1848](#),
[1856](#), [1857](#), [1859](#), [1861](#), [1882](#), [1885](#), [1889](#), [1909](#), [1912](#),
[1916](#), [3621](#), [4672](#)
`__enumext_set_error:nn` [5340](#), [5377](#), [5379](#)
`__enumext_set_item_width:` [105](#), [3762](#), [3762](#), [3796](#)
`__enumext_set_parse:n` [5340](#), [5351](#), [5367](#)
`\l__enumext_setkey_tmpa_int` [109](#), [5344](#), [5348](#)
`\l__enumext_setkey_tmpa_seq` [109](#), [5342](#), [5352](#),
[5358](#), [5360](#), [5362](#), [5374](#)
`\l__enumext_setkey_tmpa_tl` [109](#), [5350](#), [5354](#)
`\l__enumext_setkey_tmpb_seq` [109](#), [5343](#), [5346](#),
[5350](#), [5351](#)
`\l__enumext_setkey_tmpb_tl` [109](#), [5369](#), [5371](#), [5372](#)
`\l__enumext_show_answer_bool` [2296](#), [2315](#), [2706](#),
[3100](#), [3113](#), [3154](#), [3459](#), [5041](#), [5071](#)
`__enumext_show_length:nnn` [52](#), [221](#), [221](#), [5592](#),
[5593](#), [5594](#), [5595](#), [5596](#), [5597](#), [5598](#), [5599](#), [5600](#), [5601](#),
[5607](#), [5608](#), [5609](#), [5610](#), [5611](#), [5612](#), [5613](#), [5614](#), [5615](#),
[5616](#)
`\l__enumext_show_pos_tmp_int` [118](#), [3167](#), [3170](#),
[3185](#)
`\l__enumext_show_position_bool` [2299](#), [2318](#),
[2710](#), [3101](#), [3114](#), [3174](#), [5048](#)
`\g__enumext_standar_bool` [35](#), [102](#), [22](#), [243](#), [246](#), [265](#),
[337](#), [487](#), [503](#), [1880](#), [1945](#), [1957](#), [1983](#), [1996](#), [2034](#),
[2173](#), [2187](#), [2564](#), [2577](#), [2592](#), [3641](#)
`\l__enumext_standar_bool` [102](#), [105](#), [22](#), [1630](#), [2565](#),
[3613](#), [3759](#), [4665](#)
`\l__enumext_standar_first_bool` [35](#), [102](#), [22](#), [270](#),
[1867](#), [2014](#), [2076](#), [2083](#)
`__enumext_standar_item_vii:w` [123](#), [4722](#), [4740](#),
[4742](#)
`__enumext_standar_item_viii:w` [129](#), [4966](#), [4984](#),
[4986](#)
`__enumext_standar_ref:` [45](#), [722](#), [741](#), [3561](#)
`__enumext_standar_ref:n` [714](#), [722](#), [722](#)
`\g__enumext_standar_series_tl` [46](#), [1869](#), [1870](#),
[2036](#), [2039](#)
`__enumext_standar_unknown_keys:n` [3375](#), [3379](#),
[3383](#)
`__enumext_standar_unknown_keys:nn` [3375](#), [3385](#),
[3387](#)
`__enumext_standard_ref:n` [45](#)
`\g__enumext_starred_bool` [35](#), [121](#), [22](#), [253](#), [256](#), [279](#),
[338](#), [1629](#), [1907](#), [1950](#), [1961](#), [1988](#), [2003](#), [2042](#), [2147](#),
[2193](#), [2555](#), [3011](#), [4565](#)
`\l__enumext_starred_bool` [121](#), [122](#), [127](#), [22](#), [2593](#),
[2628](#), [2634](#), [2682](#), [3614](#), [4664](#), [4691](#), [4927](#), [4931](#)
`__enumext_starred_columns_set_vii:` [4342](#),
[4342](#), [4639](#)
`__enumext_starred_columns_set_viii:` [4342](#),
[4373](#), [4902](#)
`\l__enumext_starred_first_bool` [35](#), [121](#), [22](#), [284](#),
[962](#), [975](#), [1872](#), [2023](#), [2076](#), [2083](#)
`__enumext_starred_item_vii:w` [123](#), [124](#), [4722](#),
[4739](#), [4756](#)
`__enumext_starred_item_vii_aux_i:w` [4722](#),
[4761](#), [4764](#)
`__enumext_starred_item_vii_aux_ii:w` [4722](#),
[4762](#), [4767](#), [4769](#)
`__enumext_starred_item_vii_aux_iii:w` [4722](#),
[4772](#), [4779](#)
`__enumext_starred_item_viii:w` [129](#), [4983](#), [5001](#),
[5001](#)
`__enumext_starred_item_viii_aux_i:w` [129](#),
[5001](#), [5007](#), [5010](#)
`__enumext_starred_item_viii_aux_ii:w` [129](#),
[5001](#), [5008](#), [5022](#), [5024](#)
`__enumext_starred_joined_item_vii:n` [117](#), [123](#),
[4404](#), [4404](#), [4737](#)
`__enumext_starred_joined_item_viii:n` [117](#),
[129](#), [4404](#), [4453](#), [4981](#)
`__enumext_starred_ref:` [46](#), [766](#), [798](#), [3591](#)
`__enumext_starred_ref:n` [46](#), [760](#), [766](#), [766](#)
`\g__enumext_starred_series_tl` [46](#), [1874](#), [1875](#),
[2044](#), [2047](#)
`__enumext_starred_unknown_keys:n` [3357](#), [3359](#),
[3361](#)
`__enumext_starred_unknown_keys:nn` [3357](#), [3363](#),
[3365](#)
`__enumext_start_counter:` [3776](#), [3776](#), [3795](#)
`__enumext_start_from:NNn` [47](#), [852](#), [852](#), [865](#), [887](#),
[893](#)
`\l__enumext_start_i_int` [1986](#), [1998](#), [2017](#)
`__enumext_start_item_tmp_vii:` [121](#), [4642](#), [4722](#),
[4722](#)
`__enumext_start_item_tmp_viii:` [4905](#), [4966](#),
[4966](#)
`__enumext_start_item_vii:w` [123](#), [125](#), [4748](#), [4753](#),
[4776](#), [4783](#), [4831](#), [4831](#)
`__enumext_start_item_viii:w` [129](#), [4993](#), [4998](#),
[5027](#), [5106](#), [5106](#)
`\g__enumext_start_line_tl` [35](#), [22](#), [272](#), [286](#), [343](#),
[2217](#), [2222](#), [2227](#), [2241](#), [2246](#), [2251](#)
`__enumext_start_list:nn` [37](#), [99](#), [361](#), [363](#), [3789](#),
[3927](#), [4632](#), [4895](#)
`__enumext_start_list_tag:n` [3941](#), [3967](#), [4841](#),
[5116](#)
`__enumext_start_mini_vii:` [122](#), [4502](#), [4502](#), [4683](#)
`__enumext_start_mini_viii:` [128](#), [4567](#), [4567](#),
[4950](#)
`__enumext_start_save_ans_msg:` [70](#), [71](#), [2060](#),
[2060](#), [2085](#)


```

\__enumext_start_store_level: . 102, 3635, 3635,
    3788
\__enumext_start_store_level_vii: 122, 4631,
    4694, 4694
\l__enumext_start_vii_int ... 1991, 2005, 2026
\l__enumext_start_X_int ..... 85
\__enumext_stop_item_tmp_vii: .. 121, 123, 125,
    4641, 4647, 4724, 4833
\__enumext_stop_item_tmp_viii: 128, 4904, 4910,
    4968, 5108
\__enumext_stop_item_vii: 125, 126, 4831, 4833,
    4853
\__enumext_stop_item_viii: ... 5106, 5108, 5132
\__enumext_stop_list: 37, 119, 122, 361, 364, 3706,
    3714, 3876, 3883, 4525, 4533, 4590, 4597
\__enumext_stop_list_tag:n ... 3941, 3983, 4856,
    5135
\__enumext_stop_mini_vii: 119, 122, 4502, 4521,
    4687
\__enumext_stop_mini_viii: 128, 4567, 4586, 4954
\__enumext_stop_save_ans_msg: . 70, 2060, 2065,
    2961
\__enumext_stop_start_list_tag: .. 3941, 3975,
    4843, 5118
\__enumext_stop_store_level: .. 103, 104, 3664,
    3664, 3707, 3715
\__enumext_stop_store_level_vii: .. 119, 122,
    4526, 4534, 4694, 4704
\l__enumext_store_active_bool . 31, 71, 97, 2015,
    2024, 2092, 2779, 2907, 3639, 3652, 3808, 3816, 4197,
    4696, 4706, 4917, 4933
\__enumext_store_active_keys:n . 77, 102, 2375,
    2375, 3632
\__enumext_store_active_keys_vii:n . 77, 122,
    2375, 2385, 4675
\__enumext_store_addto_prop:n 78, 89, 2451, 2451,
    2459, 2619, 2992, 5032
\__enumext_store_addto_seq:n 79, 90, 2460, 2460,
    2464, 2471, 2485, 2493, 2502, 2516, 2524, 2677, 3082
\__enumext_store_anskey_arg:n .. 82, 84, 87, 88,
    2616, 2616, 2772, 2950
\l__enumext_store_anskey_arg_tl .. 32, 82, 102,
    2625, 2630, 2632, 2637, 2644, 2647, 2657, 2662, 2665,
    2671, 2677
\__enumext_store_anskey_env:n . 88, 2901, 2905,
    2935
\l__enumext_store_anskey_env_tl .. 32, 88, 102,
    2937, 2939, 2941, 2944, 2952
\__enumext_store_anskey_safe_outer: .. 85, 87
\l__enumext_store_columns_break_bool . 2627,
    2728, 2814
\l__enumext_store_current_label_tl 31, 89, 90,
    129, 97, 2976, 2979, 2982, 2988, 2990, 2992, 3049,
    3052, 3055, 3061, 3063, 3073, 3082, 5012, 5017, 5018,
    5031, 5032, 5034
\l__enumext_store_current_opt_arg_tl . 31, 91,
    129, 97, 3092, 3097, 3104, 3110, 3117, 5020
\__enumext_store_internal_ref: .. 80, 82, 2541,
    2541, 2622
\l__enumext_store_item_join_int .. 2635, 2639,
    2731, 2817
\l__enumext_store_item_star_bool . 2642, 2733,
    2819
\l__enumext_store_item_symbol_sep_dim 2654,
    2659, 2738, 2824
\l__enumext_store_item_symbol_tl . 2645, 2649,
    2736, 2822
\l__enumext_store_keyans_item_opt_sep_v_-
    tl ..... 2986, 2988, 3059, 3061
\l__enumext_store_keyans_item_opt_sep_-
    viii_tl ..... 5015, 5017
\__enumext_store_level_close: . 79, 2465, 2489,
    3668
\__enumext_store_level_close_vii: . 79, 2496,
    2520, 4710
\__enumext_store_level_open: 79, 103, 2465, 2465,
    3647, 3660
\__enumext_store_level_open_vii: .. 79, 2496,
    2496, 4700
\g__enumext_store_name_tl . 31, 71, 97, 342, 349,
    350, 351, 352, 2068, 2094, 2216, 2221, 2226, 2240,
    2245, 2250, 2959
\l__enumext_store_name_tl . 31, 71, 72, 97, 1901,
    1904, 1928, 1931, 2019, 2028, 2063, 2072, 2073, 2094,
    2095, 2097, 2098, 2100, 2102, 2103, 2105, 2107, 2108,
    2132, 2453, 2455, 2462, 2605, 2606, 2718, 3032, 3033,
    3184, 5056
\l__enumext_store_ref_key_bool 82, 2312, 2620,
    2668, 2996, 3070
\l__enumext_store_save_key_vii_bool .. 2387,
    2417
\l__enumext_store_save_key_vii_tl 2389, 2390,
    2418, 2419, 2500, 2508, 2512, 2516
\l__enumext_store_save_key_X_bool .. 77, 118
\l__enumext_store_save_key_X_tl .... 77, 118
\l__enumext_store_upper_level_X_bool .. 118
\__enumext_storing_exec: .. 71, 2070, 2086, 2090
\__enumext_storing_set:n 70, 71, 2055, 2070, 2070
\l__enumext_the_counter_v_tl ..... 838
\l__enumext_the_counter_vii_tl ..... 776
\l__enumext_the_counter_viii_tl ..... 790
\l__enumext_the_counter_X_tl ..... 37
\__enumext_tmp:n 32, 36, 39, 45, 56, 63, 64, 71, 79, 84,
    85, 96, 127, 134, 159, 163, 170, 190, 619, 628, 1808,
    1819, 2051, 2059, 2111, 2129, 2305, 2374, 2393, 2406,
    2543, 2550, 2551, 2572, 2585, 2588, 2599, 2998, 3005,
    3332, 3342, 3375, 3382, 3532, 3571, 3572, 3605
\__enumext_tmp:nn 629, 650, 651, 685, 686, 701, 882,
    907, 988, 1010, 1011, 1031, 1085, 1093, 1094, 1108,
    1173, 1189, 1190, 1203, 1697, 1713, 2270, 2304, 3316,
    3331
\__enumext_tmp:nnn 702, 718, 719, 720, 721, 748, 764,
    765
\__enumext_tmp:nnnnn 908, 933, 936, 939, 941, 943,
    946, 949
\__enumext_tmp:w ..... 5177, 5180
\l__enumext_tmpa_vii_int 4352, 4355, 4364, 4395
\l__enumext_tmpa_viii_int ..... 4383, 4386
\l__enumext_tmpa_X_dim ..... 170
\l__enumext_tmpa_X_int ..... 170
\l__enumext_topsep_v_skip 1270, 1274, 1473, 4190
\l__enumext_topsep_vii_skip .. 1550, 1559, 1563
\l__enumext_topsep_viii_skip . 1572, 1594, 1598
\__enumext_unskip_unkern: .. 35, 227, 227, 1323,
    1495, 3709, 3710, 3750, 3878, 3879, 3896, 4847, 4848,
    5122, 5123
\l__enumext_vspace_a_star_v_bool ..... 1746
\l__enumext_vspace_a_star_vii_bool ... 1768
\l__enumext_vspace_a_star_viii_bool ... 1779
\l__enumext_vspace_a_star_X_bool ..... 85

```

_enumext_vspace_above:	64, 104, 1714, 1714, 3720
_enumext_vspace_above_v:	64, 1742, 1742, 3832
\l_enumext_vspace_above_v_skip	1744, 1748, 1750
_enumext_vspace_above_vii:	65, 122, 1764, 1764, 4680
\l_enumext_vspace_above_vii_skip	1766, 1770, 1772
_enumext_vspace_above_viii:	65, 1764, 1775, 4948
\l_enumext_vspace_above_viii_skip	1777, 1781, 1783
\l_enumext_vspace_b_star_v_bool	1757
\l_enumext_vspace_b_star_vii_bool	1790
\l_enumext_vspace_b_star_viii_bool	1801
\l_enumext_vspace_b_star_X_bool	85
_enumext_vspace_below:	64, 105, 1728, 1728, 3758
_enumext_vspace_below_v:	64, 1753, 1753, 3905
\l_enumext_vspace_below_v_skip	1755, 1759, 1761
_enumext_vspace_below_vii:	65, 122, 1786, 1786, 4690
\l_enumext_vspace_below_vii_skip	1788, 1792, 1794
_enumext_vspace_below_viii:	65, 1786, 1797, 4956
\l_enumext_vspace_below_viii_skip	1799, 1803, 1805
_enumext_widest_from:nNNn	47, 866, 866, 881, 900
\g_enumext_widest_label_tl	30, 42, 52, 607, 611, 615
\l_enumext_wrap_label_opt_v_bool	3405
\l_enumext_wrap_label_opt_vii_bool	123, 4747
\l_enumext_wrap_label_opt_viii_bool	129, 4992
\l_enumext_wrap_label_opt_X_bool	85
\l_enumext_wrap_label_v_bool	3401, 3405, 3412, 3458, 3466, 4240
\l_enumext_wrap_label_vii_bool	123, 4747, 4751, 4759, 4823
\l_enumext_wrap_label_viii_bool	129, 4992, 4996, 5005, 5070, 5079
\l_enumext_wrap_label_X_bool	85
_enumext_wrapper_label_v:n	3464, 3468, 4259
_enumext_wrapper_label_vii:n	4825
_enumext_wrapper_label_viii:n	5077, 5081
\l_enumext_write_anskey_env_bool	32, 102, 2830, 2855
\l_enumext_write_anskey_env_file_iow	32, 102, 2880, 2881, 2882
\l_enumext_write_anskey_env_file_name_-tl	32, 102, 2831, 2941
\l_enumext_write_aux_file_tl	33, 81, 90, 156, 2608, 2614, 3039, 3045
enumext*	5, 4626
enumXi	570
enumXii	570
enumXiii	570
enumXiv	570
enumXv	570
enumXvi	570
enumXvii	570
enumXviii	570

Environments provide by enumext:

anskey*	29, 31, 32, 34, 71, 76, 77, 80, 81, 83, 86, 102, 103, 122, 132, 133, 138, 141
enumext*	29, 30, 33-35, 39-43, 45, 46, 48, 50, 52-54, 60, 61, 65-68, 70-73, 76-82, 84, 88, 89, 95, 96, 99, 101-103, 108, 116, 117, 119, 120, 122, 124-128, 130-134, 136, 140, 143, 144
enumext	29, 30, 34, 35, 39-50, 52-56, 59, 62-68, 70-73, 76-82, 84, 88, 89, 93-95, 97, 99, 103, 105, 106, 111, 116, 119, 121, 122, 124, 127, 132-134, 136, 139, 141, 143
keyans*	29-31, 33-36, 39-42, 45-48, 50, 52-54, 60, 61, 65, 71, 72, 75, 76, 78, 87, 89, 91, 96, 99, 101, 108, 117, 118, 127, 140, 142, 144
keyanspic	29-33, 36, 42, 46, 71, 72, 75, 78, 79, 87, 89-91, 96, 108-113, 115, 142
keyans	29-31, 33, 35, 36, 39, 40, 42, 43, 46, 48, 50, 52, 53, 56, 59, 62-64, 71, 72, 75, 76, 78, 79, 87, 89-92, 96-99, 106, 108, 110-112, 114, 119, 128, 140, 142

Environments:

center	116
description	95, 116
enumerate	116
flushleft	116
flushright	116
itemize	116
list	34, 37, 84, 95, 99, 104, 105, 108, 110, 112, 116, 119
lrbox	125
minipage	34, 37-39, 41, 54, 56, 57, 110, 113, 115, 116, 119, 125, 126
multicols	54-57, 62, 103, 104
quotation	116
quote	116
tabbing	116
trivlist	116
verbatim	116
verse	116

exp commands:

\exp_after:wN	5180
\exp_args:Ne	2949, 3629, 5168
\exp_args:NV	2744, 2841, 3345, 3363, 3385, 5465
\exp_not:N	43, 610, 736, 779, 793, 841, 1041, 1044, 1055, 1056, 1057, 1068, 1069, 1080, 1081, 2673, 2715, 2716, 3075, 3181, 3182, 5053, 5054, 5177
\exp_not:n	274, 288, 301, 309, 317, 676, 696, 736, 737, 779, 793, 841, 1042, 1835, 1844, 2283, 2332, 2436, 2449, 2611, 2639, 2649, 2659, 2673, 2674, 3042, 3077, 3079, 4054, 5294, 5304, 5497, 5502

F

\fbox	2339
\fboxrule	2339
\fboxsep	2339
file commands:	
\file_if_exist:nTF	2857
\file_input_stop:	5917
first	1094
font	629
\footnote	39
\footnote	39, 427, 457
\footnotemark	437, 467
\footnotesize	2716, 3182, 5054
\footnotetext	423
force-eol	2812
\foreachkeyans	18, 137, 5430

G

\getkeyans 18, 132, [5166](#)
group commands:
 \group_begin: 2714, 2759, 3180, 5052, 5235
 \group_end: 2721, 2775, 3188, 5059, 5242

H

\hbadness 4858, 5137
hbox commands:
 \hbox_overlap_left:n 2531, 3247, 4816
 \hbox_set:Nn 599, 4117
 \hbox_set_end: 4857, 5136
 \hbox_set_to_wd:Nnw 4834, 5109
\hfill 659, 664, 670, 671, 1653, 1680, 2673, 3075, 4529, 4593
hook commands:
 \hook_gput_code:nnn 5, 201, 205, 209, 372
 \hook_gset_rule:nnnn 373
\hyperlink 82, 90
\hyperlink 2673, 3075
\hypertarget 38
\hypertarget 398

I

\IfDocumentMetadataT 3969, 3977, 3985, 4021, 4029, 4037, 4141, 4150, 4158, 4165, 4170, 4218, 4227, 4317, 4325, 4527, 4638, 4646, 4792, 4901, 4909
\IfDocumentMetadataTF .. 489, 505, 519, 532, 3264, 3440, 4591
\IfHyperBoolean 380
\IfPackageLoadedT 376
\IfPackageLoadedTF 7, 388
\ignorespaces .. 1044, 1057, 1069, 1081, 4130, 4643, 4720, 4753, 4776, 4783, 4829, 4849, 4906, 4964, 4998, 5027, 5104, 5124
\inputlineno 274, 288, 301, 309, 317
int commands:
 \int_add:Nn 4437, 4486
 \int_case:nn ... 1218, 1343, 2142, 2168, 2207, 2231
 \int_case:nnTF 229
 \int_compare:nNnTF .. 553, 769, 783, 800, 807, 1313, 1332, 1486, 1504, 1616, 1635, 1647, 1675, 2255, 2261, 2783, 2787, 2791, 2799, 2911, 2915, 2919, 2957, 2977, 3015, 3020, 3025, 3050, 3165, 3611, 3622, 3644, 3657, 3673, 3688, 3703, 3744, 3817, 3821, 3849, 3874, 3890, 4089, 4201, 4205, 4407, 4417, 4433, 4456, 4466, 4482, 4656, 4660, 4698, 4708, 4860, 4872, 4922, 4934, 5139, 5151, 5348, 5480
 \int_compare_p:nNn ... 244, 254, 266, 267, 280, 281, 1622, 1623, 2148, 2174, 2556, 2566, 2578, 2579, 2594, 2635, 3654
 \int_decr:N 4436, 4485
 \int_eval:n .. 359, 895, 2455, 2606, 2716, 3033, 3182, 3780, 3920, 4425, 4474, 4637, 4900, 5054
 \int_from_alph:n 860, 874
 \int_from_roman:n 862, 876
 \int_gadd:Nn 4438, 4487
 \int_gdecr:N 2151, 2156, 2160, 2164, 2177
 \int_gincr:N 1985, 1990, 2618, 3085, 3201, 3235, 3418, 3733, 3841, 4245, 4726, 4802, 4970, 5036
 \int_gset:Nn 435, 465, 2200
 \int_gset_eq:NN .. 432, 462, 1884, 1891, 1897, 1903, 1911, 1918, 1924, 1930
 \int_gzero:N . 330, 331, 332, 1661, 1688, 2267, 3749, 3895, 4883, 5163
 \int_if_exist:NNTF 1859, 1895, 1901, 1922, 1928, 2105

 \int_incr:N 2798, 3167, 3610, 3812, 4088, 4655, 4725, 4921, 4969
 \int_mod:nn 4874, 5153
 \int_new:N 16, 17, 18, 19, 20, 21, 46, 47, 72, 89, 111, 125, 137, 138, 149, 150, 151, 153, 164, 165, 173, 174, 175, 176, 177, 1861, 2108
 \int_set:Nn 856, 860, 862, 1998, 2005, 2017, 2026, 4311, 4312, 4352, 4383, 4406, 4412, 4428, 4455, 4461, 4477, 4858, 5137, 5344, 5482
 \int_set_eq:NN 1986, 1991, 4435, 4484
 \int_sign:n 2202
 \int_step_function:nnN 2572, 2585, 2599
 \int_step_function:nnnN 5486
 \int_step_inline:nn 5396
 \int_step_inline:nnn 4313
 \int_to_roman:n 213, 2552, 2589
 \int_use:N 352, 357, 358, 1314, 1333, 1648, 2000, 2007, 2019, 2028, 3566, 3630, 3674, 3683, 3698, 3704, 3780, 3920, 4410, 4411, 4423, 4459, 4460, 4472, 4637, 4900, 5833, 5837, 5843, 5847
 \int_zero:N 3170, 4864, 5143

iow commands:

 \iow_char:N 2938, 2939
 \iow_close:N 2882
 \iow_new:N 106
 \iow_now:Nn 2881
 \iow_open:Nn 2880
\item . 93, 97, 123, 125, 128, 131, 365, 2473, 2479, 2504, 2510, 2632, 3052, 3055, 3253, 3422, 4145, 4146, 4640, 4642, 4903, 4905, 5034
\item* 5, 16, 75, [3420](#)
item-join [2726](#), [2812](#)
item-pos* [2726](#), [2812](#), [3316](#)
item-star [2726](#), [2812](#)
item-sym* [2726](#), [2812](#), [3316](#)
\itemindent 100
\itemindent 99
itemindent [988](#)
\itemsep 4134
\itemwidth . 569, 2339, 3764, 3770, 3909, 3915, 4446, 4450, 4495, 4499

K

keyans 15, [3922](#)
keyans* 15, [4890](#)
keyanspic 16, [4136](#)

Keys for \anskey provide by [enumext](#):

 break-col 82, 84
 force-eol 85
 item-join 82, 84
 item-pos* 82, 84
 item-star 82, 84
 item-sym* 82, 84
 overwrite 85
 write-env 85

Keys for anskey* provide by [enumext](#):

 break-col 82, 84
 force-eol 85
 item-join 82, 84
 item-pos* 82, 84
 item-star 82, 84
 item-sym* 82, 84
 overwrite 85
 write-env 85

Keys for environments provide by [enumext](#):

above*	31, 49, 63–65, 104, 122
above	31, 49, 63–65, 104, 122, 128
after	52, 105, 122, 128
align	31, 43, 91, 93, 94, 98, 124, 138
base-fix	49, 66, 78, 102
before*	52, 104, 122, 128
before	52
below*	31, 63–65, 105, 122
below	31, 63–65, 105, 122, 128
check-ans	33–35, 70–75, 78, 88, 90, 105, 106, 122, 126, 140
columns-sep	53, 103, 126
columns	31, 53, 63, 103
first	52, 125
font	43, 94, 98, 114, 124
item-pos*	93, 95
item-sym*	32, 93, 95
itemindent	31, 50, 51, 93, 97–99, 125
itemsep	48, 101, 126
label-pos	110, 112, 114
label-sep	110
labelsep	43, 100, 124
labelwidth	42–46, 48, 100, 124
label	30, 42, 44, 47, 48, 112, 116
layout-sep	110
layout-sty	110, 115, 116
layout-top	110
lisparindent	101
list-indent	30, 50, 112
list-offset	50, 105, 108
listparindent	50, 125
mark-ans*	75, 78, 92
mark-ans	76, 78, 83
mark-pos*	75, 78, 92
mark-pos	32, 76, 78, 138
mark-ref	76, 78, 80, 82
mark-sep*	75, 78, 91, 92
mark-sep	32, 76, 78, 130
mini-env	31, 39–41, 53, 62, 63, 78, 104, 116, 119, 120, 122, 128
mini-right*	31, 34, 54, 78, 119, 120, 122
mini-right	31, 34, 54, 61, 78, 119, 120, 122
mini-sep	31, 53, 78, 104
mode-box	43, 93–95, 98, 99
no-store	33, 70–72, 78, 84, 87, 93
noitemsep	48
nosep	48
overwrite	32, 86
parindent	101
parsep	48, 101, 112, 125
partopsep	48
ref	30, 44–46, 99, 140
resume*	30, 65, 66, 70, 71, 78, 105, 122, 134
resume	30, 37, 65–71, 78, 105, 122, 134, 135
rightmargin	50, 116
save-ans	31, 37, 66–73, 77–79, 84, 85, 87–90, 97, 106, 113, 124, 127, 128, 130, 132, 135, 140
save-key	32, 66, 77, 78, 102, 122
save-pos	78
save-ref	33, 38, 76, 78, 80, 82, 89, 90, 97, 130
save-sep	75, 78, 89, 129
series	30, 65–70, 78, 102, 105, 122, 135
show-ans	32, 75, 76, 78, 80, 82, 83, 91, 92, 114, 130
show-length	35, 52, 99, 139, 140

show-pos	32, 75, 76, 80, 82, 83, 91, 114, 130
start*	31, 47, 48, 66
start	31, 34, 47, 48, 66
store-key	77
topsep	48, 49, 112
widest	30, 34, 47, 48
wrap-ans*	33, 75, 78, 98, 114
wrap-ans	41, 76, 78, 80, 83
wrap-label*	31, 43, 93, 94, 97, 98, 123, 124, 129
wrap-label	31, 43, 93, 94, 97, 98, 112, 114, 123, 124, 129
wrap-opt	75, 78, 91, 97, 114
wrap-sep	83
write-env	32, 86

keys commands:

\keys_define:nn	621, 631, 653, 688, 704, 750, 815, 884, 910, 952, 990, 1013, 1087, 1096, 1175, 1192, 1699, 1810, 2053, 2113, 2272, 2307, 2395, 2400, 2726, 2812, 3318, 3334, 3357, 3377, 4043, 5194, 5306, 5422, 5430
\keys_if_exist_p:nn	5418, 5419
\l_keys_key_str	84, 86, 2744, 2841, 3345, 3363, 3385, 5465, 5577
\keys_precompile:nnN	133, 196, 196, 5196, 5202, 5208, 5214, 5220, 5226, 5448
\keys_set:nn	645, 969, 981, 1198, 1704, 1709, 1947, 1952, 2039, 2047, 2343, 2344, 2348, 2349, 2353, 2354, 2358, 2359, 2363, 2364, 2368, 2369, 2764, 2893, 3624, 3629, 3828, 4061, 4063, 4065, 4067, 4069, 4071, 4073, 4075, 4077, 4079, 4099, 4673, 4943, 5310, 5315, 5316, 5317, 5318, 5321, 5326, 5327, 5328, 5329, 5330, 5331, 5332, 5364, 5474

keyval commands:

\keyval_parse:NNn	1824, 2425, 5282
-------------------	------------------

L

label	702, 748, 815
label-pos	4043
label-sep	4043

Labels provide by [enumext](#):

\Alph*	42
\Roman*	42
\alph*	42
\arabic*	42
\roman*	42
labelsep	629
\labelwidth	42
labelwidth	629
\lastnodetype	229
layout-sep	4043
layout-sty	4043
layout-top	4043
\leftmargin	100
\leftmargin	99, 4129

legacy commands:

\legacy_if:nTF	4787, 4790, 5087, 5090
\legacy_if_gset_false:n	559, 4542
\legacy_if_set_false:n	4789, 5089
\legacy_if_set_true:n	4752, 4775, 4782, 4796, 4997, 5026
\linewidth	104
\linewidth	3728, 3764, 3838, 3909, 4310, 4355, 4386, 4508, 4573
\list	363
list-indent	988
list-offset	988
\listparindent	4132

listparindent	988	\multicolsep	1317, 1489, 3694, 3865
M		N	
\makebox	116	\NeedsTeXFormat	3
\makebox	2533, 3300, 3490, 4235, 4248, 4820, 5099	\NewCommandCopy	365
\makelabel	93, 94, 98, 116	\newcounter	576
\makelabel	93, 97, 3280, 3296, 3474, 3486	\NewDocumentCommand	1614, 2756, 4195, 5166, 5233, 5340, 5389, 5467
mark-ans	2305, 4043	\NewDocumentEnvironment	2886, 3783, 3922, 4136, 4626, 4890
mark-ans*	2270, 2305	\newlabel	38
mark-pos	2305, 4043	\newlabel	410
mark-pos*	2270, 2305	no-store	2111
mark-ref	2305	\noindent	3735, 4517, 4582, 4863, 5142
mark-sep	2305, 4043	\nointerlineskip	1326, 1329, 1498, 1501, 1655, 1682, 4517, 4582
mark-sep*	2270, 2305	noitemsep	908
mini-env	1173	\nopagebreak	1263, 1291, 1326, 1329, 1498, 1501, 1605, 1611
mini-sep	1173	\normalfont	2715, 3181, 5053
\minipage	369	nosep	908
\miniright	11, 62, 1614, 1665, 1692, 3747, 3893	O	
mode commands:		\obeyedline	2938, 2939
\mode_if_math:TF	2807, 2930	overwrite	2812
\mode_if_vertical:TF	1252, 1280, 1300, 1324, 1475, 1496	P	
\mode_leave_vertical:	967, 978, 1041, 1055, 2529, 3245, 4814	Packages:	
mode-box	619	caption	119
msg commands:		enumext	29, 41, 44, 70, 75, 95, 99, 100, 110, 138, 139
\msg_error:nn	1667, 1694, 2768, 2801, 2805, 2897, 2928, 3819, 3823, 4091, 4148, 4203, 4658, 4924, 4936, 5333, 5392	enumitem	42
\msg_error:nnn	727, 773, 787, 835, 1618, 1625, 1632, 1663, 1690, 1959, 1963, 2078, 2750, 2809, 2847, 2909, 2913, 2917, 2921, 2932, 3351, 3369, 3391, 4662, 4929, 5182, 5191, 5275, 5380, 5411, 5420, 5457, 5478	expl3	116
\msg_error:nnnn	2753, 2781, 2785, 2789, 2793, 2850, 3354, 3372, 3394, 3810, 4199, 4207, 4919, 5254, 5460	footnotehyper	38, 40
\msg_error:nnnnn	675, 695, 2282, 2331, 4053	hyperref	33, 34, 38, 82, 90, 124, 138
\msg_fatal:nn	3612	latex-lab-block	37
\msg_fatal:nnn	573	ltxcmd	37, 85
\msg_info:nnn	9, 12, 378, 390	ltsockets	108
\msg_line_context:	5537, 5542, 5547, 5552, 5581, 5586, 5591, 5606, 5621, 5625, 5629, 5633, 5637, 5641, 5648, 5655, 5661, 5675, 5679, 5684, 5688, 5692, 5696, 5701, 5705, 5709, 5713, 5718, 5765, 5769, 5774, 5779, 5783, 5788, 5864, 5868, 5873, 5878, 5883, 5887, 5891, 5895, 5899, 5903, 5907, 5911, 5915	lua-visual-debug	56
\msg_log:nnn	2097, 2102, 2107	multicol	29, 138
\msg_log:nnnnn	356, 2240, 2245, 2250	scontents	85
\msg_log:nnnnnn	348	shortlst	116, 121, 125
\msg_new:nnn	5505, 5509, 5513, 5517, 5522, 5535, 5539, 5544, 5549, 5554, 5563, 5571, 5575, 5579, 5584, 5589, 5604, 5619, 5623, 5627, 5631, 5635, 5639, 5643, 5652, 5658, 5664, 5668, 5672, 5677, 5682, 5686, 5690, 5694, 5699, 5703, 5707, 5711, 5716, 5751, 5755, 5759, 5763, 5767, 5772, 5777, 5781, 5786, 5862, 5866, 5871, 5876, 5881, 5885, 5889, 5893, 5897, 5901, 5905, 5909, 5913	tagpdf	108
\msg_new:nnnn	5526, 5721, 5730, 5739, 5745, 5790, 5800, 5810, 5820, 5830, 5840, 5850, 5856	\par	1263, 1291, 1329, 1501, 1605, 1611, 1650, 1655, 1677, 1682, 2681, 3711, 3880, 3898, 4181, 4184, 4330, 4544, 4559, 4605, 4619, 4863, 5142
\msg_term:nnnn	2062, 2067, 3555, 3565, 3596, 3601	para commands:	
\msg_term:nnnnn	2221	\para_end:	4880, 5160
\msg_warning:nn	3746, 3892	\parbox	2339
\msg_warning:nnn	2861, 2865, 2870	\parindent	4845, 5120
\msg_warning:nnnn	2258, 2264, 3504, 3509, 4409, 4422, 4458, 4471	\parsep	55, 111
\msg_warning:nnnnn	2216, 2226	\parsep	968, 3588, 4113, 4122
\multicolsep	103	parsep	908
		\parskip	4846, 5121
		\partopsep	3589, 3896, 4133
		partopsep	908
		peek commands:	
		\peek_meaning:NTF	4731, 4745, 4760, 4771, 4975, 4990, 5006
		\peek_meaning_remove:NTF	4738, 4982
		\peek_remove_spaces:n	3427
		\phantomsection	38
		\phantomsection	399
		prg commands:	
		\prg_do_nothing:	403
		\prg_new_protected_conditional:Npnn	215, 2853
		\prg_replicate:nn	224
		\prg_return_false:	219, 2866, 2874

\prg_return_true: 218, 2862, 2871

\printkeyans 19, 132, 5233

prop commands:

\prop_const_from_keyval:Nn 5381

\prop_count:N 350, 2455, 2606, 2718, 3033, 3184, 5056, 5483

\prop_get:NnNTF 5407

\prop_gput_if_not_in:Nnn 2453

\prop_if_exist:NTF 2095, 5186, 5476

\prop_item:Nn 5188, 5500

\prop_new:N 2098

\ProvidesExplPackage 4

R

\raggedcolumns 3697, 3868

\raisebox 4272

\ref 80, 89

ref 702, 748, 815

\refstepcounter 4799, 5092

regex commands:

\regex_if_match:nnTF 217, 859, 861, 873, 875

\renewcommand 736, 779, 793, 841

\RenewDocumentCommand . 427, 457, 1665, 1692, 2938, 3253, 3280, 3296, 3422, 3474, 3486, 4146

\RequirePackage 13

resume 1808

resume* 1808

rightmargin 988

\Roman 42, 47, 48

\Roman 595

\roman 42, 47, 48

\roman 596, 720, 5218

S

save-ans 2051

save-key 2393

save-ref 2305

save-sep 2270, 2305, 4043

scan commands:

\scan_stop: 4145, 4640, 4903, 5177, 5180

seq commands:

\seq_clear:N 5342, 5485

\seq_const_from_clist:Nn 5335

\seq_count:N 351, 4336, 5346

\seq_gclear:N 452, 453, 482, 483

\seq_gput_right:Nn 438, 439, 468, 469, 2462

\seq_if_empty:NTF 445, 475, 5248, 5360

\seq_if_exist:NTF 2100, 5246

\seq_if_in:NnTF 5252

\seq_item:Nn 4323

\seq_map_function:NN 5351

\seq_map_inline:Nn 5261, 5269, 5361, 5362

\seq_map_pairwise_function:NNN 447, 477

\seq_new:N 112, 113, 115, 135, 166, 167, 168, 169, 2103

\seq_pop_left:NN 5350

\seq_put_right:Nn 4209, 5358, 5374, 5495

\seq_set_from_clist:Nn 5343

\seq_set_map_e:NNn 5352

\seq_use:Nn 196, 197, 5491

series 1808

\setcounter .. 870, 874, 876, 3778, 3920, 4178, 4637, 4900

\setenumext 6, 134, 5340

\setenumextmeta 6, 136, 5381

show-ans 2270, 2305, 4043

show-length 1085

show-pos 2270, 2305, 4043

skip commands:

\skip_add:Nn 1223, 1232, 1241, 1254, 1258, 1282, 1286, 1302, 1360, 1362, 1376, 1379, 1400, 1402, 1416, 1419, 1439, 1441, 1455, 1458, 1477, 1526, 1527, 1538, 1540, 4122, 4131

\skip_gset:Nn 1553, 1557, 1561

\skip_gzero_new:N 1548, 1549

\skip_horizontal:N .. 1056, 1068, 1080, 4817, 4829, 4867, 5104, 5146

\skip_horizontal:n .. 1042, 2530, 2538, 3246, 3248, 4253, 4716, 4815, 4849, 4960, 5124

\skip_if_eq:nnTF 1221, 1230, 1239, 1346, 1386, 1426, 1514, 1550, 1572, 1716, 1730, 1744, 1755, 1766, 1777, 1788, 1799

\skip_new:N ... 66, 67, 68, 73, 74, 75, 76, 77, 78, 188

\skip_set:Nn 1206, 1210, 1268, 1272, 1296, 1349, 1350, 1368, 1389, 1390, 1408, 1428, 1429, 1447, 1471, 1517, 1518, 1532, 1552, 1556, 1574, 1578, 1582, 1588, 1592, 1596, 4106

\skip_set_eq:NN 1307, 1308, 1310, 1317, 1482, 1483, 1484, 1489, 3545, 3587, 3588, 4846, 5121

\skip_sub:Nn 1356, 1358, 1372, 1374, 1396, 1398, 1412, 1414, 1435, 1437, 1451, 1453, 1524, 1525, 1536, 1537

\skip_use:N 1208, 1212, 1256, 1260, 1264, 1284, 1288, 1298, 1304, 1717, 1721, 1724, 1731, 1735, 1738, 3711

\skip_vertical:N . 560, 563, 980, 4543, 4557, 4882, 5162

\skip_vertical:n 979, 4881, 5161

\skip_zero:N 1316, 1330, 1468, 1469, 1470, 1488, 1502, 3589, 3694, 3865, 4133, 4134

\skip_zero_new:N 1547, 1569, 1570, 1571

\c_zero_skip . 560, 563, 980, 1221, 1230, 1239, 1387, 1426, 1550, 1572, 1717, 1731, 1744, 1755, 1766, 1777, 1788, 1799, 4543, 4557, 4882, 5162

\small 5201, 5207, 5213, 5219, 5225, 5231

\smash 3298, 3488

socket commands:

\socket_assign_plug:nn .. 3971, 3979, 3987, 4023, 4031, 4039

\socket_new:nn 3941, 3991

\socket_new_plug:nnn 3942, 3950, 3958, 3992, 4000, 4009

\socket_use:n 4024, 4032, 4040

\socket_use:nn 3972, 3980, 3988

start 882

start* 882

start-list-tags 3941, 3991

\stepcounter 431, 461, 4116, 4265

stop-list-tags 3941, 3991

stop-start-tags 3941, 3991

str commands:

\c_backslash_str 2809, 5542, 5547, 5552, 5557, 5559, 5561, 5566, 5568, 5666, 5670, 5674, 5684, 5688, 5696, 5697, 5701, 5713, 5714, 5718, 5719, 5740, 5742, 5746, 5748, 5788, 5851, 5853, 5857, 5859, 5868, 5869, 5873, 5878, 5879, 5883, 5887, 5891

\c_circumflex_str 108

\c_colon_str 2605, 3032, 5177

\c_left_brace_str 5647, 5654, 5660

\c_percent_str 108

\c_right_brace_str 5647, 5654, 5660

\str_case:nn 237, 294, 3125

\str_case:nnTF . 1831, 1839, 2432, 2440, 5289, 5298

\str_clear:N 3621, 4672

<code>\str_const:Nn</code>	107
<code>\str_count:n</code>	224
<code>\str_if_empty:NTF</code>	1848, 1889, 1916
<code>\str_if_eq:nnTF</code>	3547, 3592, 5391
<code>\str_if_in:nnTF</code>	5173
<code>\str_new:N</code>	69, 120, 121, 122, 140, 183
<code>\str_set:Nn</code>	660, 666, 672, 691, 692, 693, 2278, 2279, 2280, 2327, 2328, 2329, 4048, 4051
<code>\str_set_eq:NN</code>	3148, 5044, 5061
<code>\str_use:N</code>	3302
<code>\strut</code>	3298, 3488
<code>\strutbox</code>	1335, 1338, 1349, 1350, 1361, 1363, 1378, 1381, 1389, 1390, 1401, 1403, 1418, 1421, 1428, 1429, 1440, 1442, 1457, 1460, 1506, 1509, 1517, 1518, 1526, 1527, 1539, 1541, 1552, 1553, 1556, 1563, 1576, 1584, 1590, 1598, 4125, 4131, 4181, 4189, 4278

T

tag commands:	
<code>\tag_mc_begin:n</code>	3948, 3998, 4007
<code>\tag_mc_begin_pop:n</code>	3964, 4016, 4173, 4175
<code>\tag_mc_end:</code>	3952, 4002, 4011
<code>\tag_mc_end_push:</code>	3945, 3995, 4161
<code>\tag_resume:n</code>	3944, 3994, 4152, 4160, 4229, 4327, 4527, 4591
<code>\tag_struct_begin:n</code>	3946, 3947, 3954, 3955, 3956, 3996, 3997, 4004, 4005, 4006, 4162
<code>\tag_struct_end:n</code>	3953, 3960, 3961, 3962, 3963, 4003, 4012, 4013, 4014, 4015, 4172, 4174, 4646, 4909
<code>\tag_suspend:n</code>	3965, 4017, 4143, 4154, 4167, 4220, 4319, 4638, 4901
<code>\tag_tool:n</code>	4153

TeX and \TeX_{ϵ} commands:

<code>\@auxout</code>	408
<code>\@currentenv</code>	237, 294
<code>\protected@write</code>	408

tex commands:

<code>\tex_scantokens:D</code>	198
--------------------------------	-----

text commands:

<code>\text_expand:n</code>	5169
<code>\textasteriskcentered</code>	2275, 2322
<code>\textborn</code>	3322
<code>\textreferencemark</code>	2310
<code>\thepage</code>	414

tl commands:

<code>\c_space_tl</code>	3104, 3117, 5591, 5606, 5629, 5633, 5832, 5833, 5842, 5843, 5903, 5907
<code>\tl_clear:N</code>	658, 665, 2268, 2379, 2389, 2410, 2418, 2625, 2976, 3049, 5012
<code>\tl_clear_new:N</code>	605
<code>\tl_const:Nn</code>	589
<code>\tl_gclear:N</code>	342, 343, 344, 1869, 1874, 3291, 3311, 4563, 4623, 4818
<code>\tl_gclear_new:N</code>	1856
<code>\tl_gput_right:Nn</code>	590
<code>\tl_greplace_all:Nnn</code>	611
<code>\tl_gset:Nn</code>	271, 272, 285, 286, 1857, 1870, 1875, 2094, 3222, 4766
<code>\tl_gset_eq:NN</code>	607, 3218, 4811
<code>\tl_if_blank:nTF</code>	2748, 2766, 2845, 2895, 3349, 3367, 3389, 4809, 5455
<code>\tl_if_empty:NTF</code>	725, 743, 771, 785, 802, 809, 833, 847, 1882, 1887, 1909, 1914, 1972, 2036, 2044, 2073, 2132, 2469, 2500, 2645, 2959, 2986, 3059, 3097, 3110, 3243, 4334, 5015, 5372
<code>\tl_if_empty:nTF</code>	1937
<code>\tl_if_exist:NTF</code>	1942

<code>\tl_if_novalue:nTF</code>	429, 459, 2762, 2891, 2984, 3057, 3090, 3197, 3216, 3224, 3399, 3619, 4097, 4670, 4941, 5013
<code>\tl_map_inline:Nn</code>	608
<code>\tl_new:N</code>	29, 30, 31, 34, 37, 38, 41, 42, 48, 50, 51, 53, 54, 90, 91, 92, 98, 99, 100, 101, 102, 103, 105, 109, 110, 114, 116, 117, 118, 126, 129, 130, 147, 156, 157, 158, 161, 182
<code>\tl_put_left:Nn</code>	2477, 2508, 2630, 4547, 4608, 5031, 5034
<code>\tl_put_right:Nn</code>	606, 839, 2481, 2512, 2559, 2569, 2582, 2597, 2603, 2608, 2632, 2637, 2644, 2647, 2657, 2662, 2665, 2671, 2944, 2979, 2982, 2988, 2990, 3017, 3022, 3027, 3030, 3039, 3052, 3055, 3061, 3063, 3073, 5017, 5018
<code>\tl_remove_all:Nn</code>	5371
<code>\tl_remove_once:Nn</code>	2547, 3002
<code>\tl_replace_all:Nnn</code>	610, 2939, 5406
<code>\tl_reverse:N</code>	2546, 2548, 3001, 3003
<code>\tl_set:Nn</code>	43, 241, 251, 298, 299, 306, 307, 314, 315, 575, 659, 664, 670, 671, 724, 734, 768, 777, 791, 832, 1039, 1053, 1066, 1078, 1971, 2072, 2380, 2390, 2411, 2419, 2712, 2831, 2937, 3092, 3178, 3337, 4082, 5020, 5050, 5369, 5405, 5475
<code>\tl_set_eq:NN</code>	616, 730, 776, 790, 838, 2545, 3000, 3013, 3146, 5043
<code>\tl_to_str:n</code>	1942, 1948, 1953, 5169
<code>\tl_trim_spaces:n</code>	606, 5358, 5369, 5375, 5391
<code>\tl_use:N</code>	612, 615, 745, 804, 811, 849, 1111, 1115, 1119, 1123, 1127, 1131, 1135, 1139, 1143, 1147, 1151, 1155, 1159, 1163, 1167, 1171, 2535, 2552, 2560, 2571, 2584, 2589, 2600, 3205, 3211, 3239, 3282, 3284, 3290, 3305, 3402, 3406, 3413, 3476, 3479, 3481, 3494, 3790, 3928, 4250, 4258, 4554, 4615, 4822, 4850, 4851, 5101, 5125, 5130, 5236, 5237, 5238, 5239, 5240, 5257, 5354, 5473

token commands:

<code>\token_to_str:N</code>	410
<code>\topsep</code>	3896, 4131
<code>topsep</code>	908
<code>\topskip</code>	1316, 1488

U

<code>\unkern</code>	232
<code>unknown</code>	2726, 2812, 3332, 3357, 3375
<code>\unskip</code>	231
use commands:	
<code>\use:N</code>	225, 3287, 3308, 3792
<code>\use:n</code>	1822, 2423, 5175, 5280
<code>\use_none:nn</code>	402, 5412
<code>\usecounter</code>	3546, 3590

V

<code>\value</code>	1885, 1891, 1898, 1904, 1912, 1918, 1925, 1931
vbox commands:	
<code>\vbox_set:Nn</code>	4222
<code>\vbox_set_top:Nn</code>	4552, 4613
<code>\vspace</code>	968, 1721, 1724, 1735, 1738, 1748, 1750, 1759, 1761, 1770, 1772, 1781, 1783, 1792, 1794, 1803, 1805

W

<code>widest</code>	882
<code>wrap-ans</code>	2305
<code>wrap-ans*</code>	2270, 2305, 4043
<code>wrap-label</code>	629
<code>wrap-label*</code>	629
<code>wrap-opt</code>	2270, 2305, 4043
<code>write-env</code>	2812