

enumext

ENUMERATE EXERCISE SHEETS

V1.0 2024-11-01^{*}

©2024 by Pablo González[†]

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

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

Abstract

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

Contents

1	Introduction	1	6	The storage system	11
1.1	Description and usage	2	6.1	Keys for storage system	11
1.2	The concept of left margin	3	6.1.1	Keys for label and ref	12
1.3	User interface	3	6.1.2	Keys for wrap and display	12
1.3.1	Internal counters	3	6.1.3	Keys for debug and checking	12
1.3.2	Public dimension	3	6.2	The command <code>\anskey</code>	13
1.3.3	Support for <code>multicol</code>	4	6.2.1	Keys for <code>\anskey</code>	13
1.3.4	Support for <code>minipage</code>	4	6.3	The environment <code>anskey*</code>	13
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>	14
2	The environments provided	5	6.4.1	The <code>\item*</code> in <code>keyans</code>	15
2.1	The environment <code>enumext</code>	5	6.5	The environment <code>keyanspic</code>	15
2.2	The environment <code>enumext*</code>	5	6.5.1	Keys for <code>keyanspic</code>	16
2.3	The command <code>\item*</code>	5	6.5.2	The command <code>\anspic</code>	16
2.3.1	Keys for <code>\item*</code>	6	6.6	Printing stored content	17
2.4	The command <code>\item</code> in <code>enumext*</code>	6	6.6.1	The command <code>\getkeyans</code>	17
3	The command <code>\setenumext</code>	6	6.6.2	The command <code>\foreachkeyans</code>	17
4	The command <code>\setenumextmeta</code>	6	6.6.3	The command <code>\printkeyans</code>	18
5	The <code>keyval</code> system	7	7	Full examples	19
5.1	Keys for <code>label</code> and <code>ref</code>	7	8	Tagged PDF examples	21
5.2	Keys for spaces	8	9	The way of non-enumerated lists	22
5.2.1	Vertical spaces	8	10	References	24
5.2.2	Horizontal spaces	9	11	Change history	24
5.3	Keys for add code	9	12	Index of Documentation	25
5.4	Keys for <code>start</code> , <code>series</code> and <code>resume</code>	10	13	Implementation	27
5.5	Keys for <code>multicol</code> s	10	14	Index of Implementation	142
5.6	Keys for <code>minipage</code>	11			
5.6.1	The command <code>\miniright</code>	11			
5.6.2	The key <code>mini-right</code>	11			

Motivation and acknowledgments

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

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

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

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

^{*}This file describes a documentation for v1.0, last revised 2024-11-01.

[†]E-mail: pablgonz@educarchile.cl.

License and Requirements

Permission is granted to copy, distribute and/or modify this software under the terms of the LaTeX Project Public License (lpp), version 1.3 or later (<https://www.latex-project.org/lpp1.txt>). The software has the status “maintained”.

The `enumext` package loads and requires `multicol`[3] and `scontents`[4] packages, need to have a modern TeX distribution such as TeX Live or MiKTeX. It has been tested with the standard classes provided by L^AT_EX: `book`, `report`, `article` and `letter` on 10pt, 11pt and 12pt.

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

1 Introduction

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

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

- Factor $x^2 - 2x + 1$
- Factor $3x + 3y + 3z$
- True False
 - $\alpha > \delta$
 - L^AT_EXze is cool?
- Related to Linux
 - You use linux?
 - Usually uses the package manager?
 - Rate the following package and class
 - `xsim-exam`
 - `xsim`
 - `exsheets`

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

- Factor $x^2 - 2x + 1$
*
- Factor $3x + 3y + 3z$
*
- True False
 - $\alpha > \delta$
*
 - L^AT_EXze is cool?
*
- Related to Linux
 - You use linux?
*
 - Usually uses the package manager?
*
 - Rate the following package and class
 - `xsim-exam`
*
 - `xsim`
*
 - `exsheets`
*

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”:

- $(x-1)^2$ *
- $3(x+y+z)$ *
- False *
 - Very True! *
- Yes *
 - Yes, dnf *
 - doesn’t exist for now :(*
 - very good *
 - obsolete *

Another very common thing to use in my work is “multiple choice questions”, for example:

1. First type of questions

- | | |
|------------|----------|
| A) value | C) value |
| B) correct | D) value |

2. Second type of questions

- $2\alpha + 2\delta = 90^\circ$
- $\alpha = \delta$
- $\angle EDF = 45^\circ$

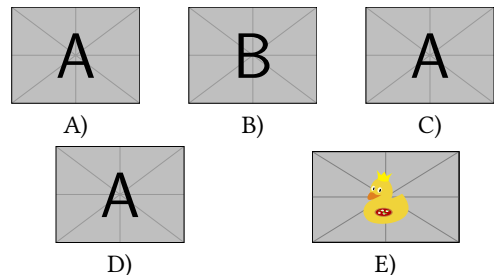
- | | |
|------------------|-------------------|
| A) I only | D) I and III only |
| B) II only | E) I, II, and III |
| C) I and II only | |

* 3. Third type of questions

- $2\alpha + 2\delta = 90^\circ$
- $\angle EDF = 45^\circ$

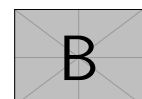
- | | |
|----------|----------|
| A) value | D) value |
| B) value | E) value |
| C) value | |

4. Question with image and label below:



5. Question with image on left side:

- value
- value
- value
- correct
- value



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

- | | | |
|-----------------|----------------------|---|
| 1. B) $x = 5$ | ※ 4. E) A duck | ※ |
| 2. D) | ※ 5. D) “other note” | ※ |
| 3. C) some 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 `MiKlTeX`, 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`.

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

The package is loaded in the usual way:

```
\usepackage{enumext}
```

1.2 The concept of left margin

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

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

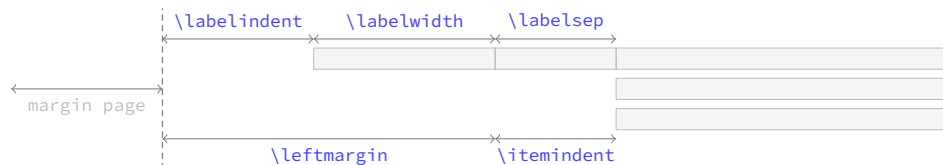


Figure 1: Representation of horizontal lengths in `enumitem`.

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

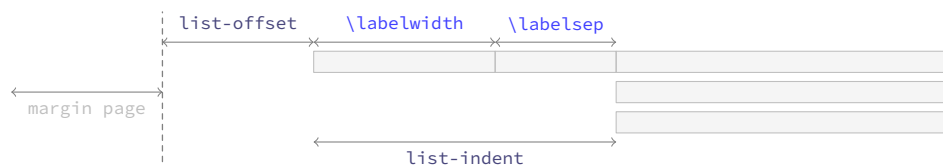


Figure 2: Representation of horizontal lengths concept in `enumext`.

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

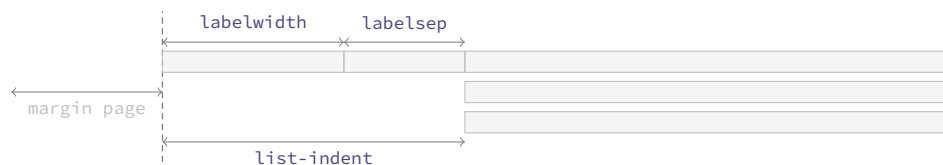


Figure 3: Default horizontal lengths `list-offset=0pt`, `list-indent=\labelwidth+\labelsep` in `enumext`.

1.3 User interface

The user interface consists of two main list environments `enumext` (vertical) and `enumext*` (horizontal), the environment `anskey*` and the command `\anskey` to “store content” and the environments `keyans`, `keyans*` and `keyanspic` for multiple choice. It also provides the commands `\getkeyans` to print individual *stored content*, `\printkeyans` 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.

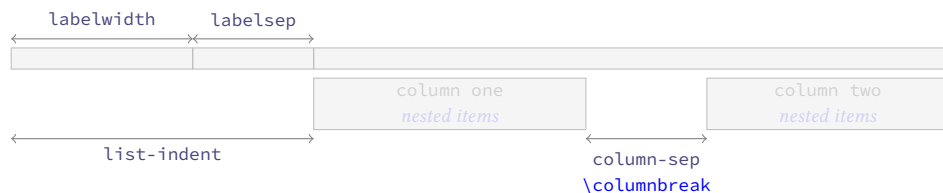


Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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:

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

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

2 The environments provided

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

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

2.1 The environment `enumext`

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

Example with `columns=2`

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

2.2 The environment `enumext*`

The `enumext*` is a *horizontal list environment* similar to the `shortenumerate` or `tasks` environments provided by the `shortlst`[15] and `tasks`[16] packages, `\item` and `\item[⟨custom⟩]` work as usual. The options can be configured globally using `\setenumext` command and locally using `[⟨key = val⟩]` in the environment.

Some considerations to take into account for this environment:

- The environment cannot be nested within itself or in the environment `keyans*`, but it can be nested within `enumext` and vice versa.
- Each “*item content*” in the environment is placed within a `minipage` environment whose *width* is stored in the dimension `\itemwidth` that NOT includes `labelwidth`, `labelsep`, only the *width of the content*.
- You cannot have floating environments like `figure` or `table` but `\footnote` with `hyperref` support is supported if the `footnotehyper` package is loaded (see §1.3.6 for full support).
- You cannot have any standard list environments like `itemize`, `enumerate`, `description`, `quote`, `quotation`, `verse`, `center`, `flushleft`, `flushright`, `verbatim`, `tabbing`, `trivlist`, `list` and all environments created with `\newtheorem`.

Example with `columns=2`

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

2.3 The command `\item*`

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

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

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

2.3.1 Keys for \item*

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

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

`item-pos*` = { $\langle rigid length \rangle$ } default: by levels

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

2.4 The command \item in enumext*

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

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

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

3 The command \setenumext

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

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

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

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

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

5 The keyval system

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

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

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

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

5.1 Keys for label and ref

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

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

- This key is intended for compatibility with *tagged PDF* and is forcibly “*enabled*” when `\DocumentMetadata` is present. If you want to get the same document output whether `\DocumentMetadata` is active or not, you must enable this key.
- In the `enumext*` and `keyans*` environments `\makeLabel` are redefined using `\makebox` by default. If `enumext` or `keyans` is used in the `enumext*` environment the key must be activated manually.

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

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

- This key is intended to give the basic structure with which the $\langle label \rangle$ will be displayed, and the form in which it is used by standard “*label and ref*” and the “*internal label and ref*” system with the `save-ref` key. You cannot use commands with $\langle label \rangle$ as an argument, for example `\emph{\langle label \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 `label` key. Internally sets the value of `\labelwidth` for the current level. The default values are calculated by means of the *width* of a box by setting a *value* to the current counter using ‘`o`’ for `\arabic*`, ‘`M`’ for `\Alpha*`, ‘`m`’ for `\alpha*`, ‘`VIII`’ for `\Roman*` and ‘`viii`’ for `\roman*`.

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

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

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

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

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

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

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

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

```
\NewDocumentCommand \mywrap { s m }
{
  \IfBooleanTF{\#\}
  {\textcolor{red}{\textbf{Q}}\textcolor{blue}{\textbf{.}}\textcolor{gray}{\#\}}
  {\textcolor{blue}{\textbf{Q}}\textcolor{red}{\textbf{.}}\textcolor{gray}{\#\}}
}
```

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

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

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

`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 `enumxt` 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 `enumxt` and `enumxt*` 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 `enumxt` and `enumxt*` 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 `enumxt*` 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 `enumxt` 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 `enumxt*` 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 `enumxt` and `enumxt*` 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 `enumxt*` 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 `enumxt`. Fix the *baseline* when an environment `enumxt` is nested in `enumxt*` and there is no material between the `\item` and the start of the environment for example `\item \begin{enumxt}` within the environment `enumxt*`. 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 `enumxt*` environment it will be lost and you will need to adjust it using the `list-offset` key in the `enumxt` environment.

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

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

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

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

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

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

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

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

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

5.2.2 Horizontal spaces

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

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

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

Sets the *indentation* of the whole environment under the box defined by `labelwidth` and `labelsep` keys. Internally sets the value of `\leftmargin` and `\itemindent` for the current level. If `list-indent=0pt` is set in the environments `enumext` and `keyans` the *(label)* will be part of the text, separated by the value of the `labelsep` key and the *first word*, in simple terms it will look like a “common paragraph”.

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

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

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

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

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

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

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

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

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

5.3 Keys for add code

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

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

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

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

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

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

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

5.4 Keys for start, series and resume

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

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

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

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

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

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

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

5.5 Keys for multicols

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

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

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.

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

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.

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: \%

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

6.1.2 Keys for wrap and display

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

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

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

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

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

Displays the *argument* passed to the `\anskey`, the *body* for `anskey*` environment, the $\langle label \rangle$ for `\item*` and `\anspic*` at the place where it is executed. If the *optional argument* is present in `\item*` or `\anspic*` it will be shown using `wrap-opt` key.

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

Sets the *symbol* to be displayed in the left margin for `\anskey`, `anskey*`, `\item*` and `\anspic*` in the place where they are executed when using the key `show-ans`.

`mark-pos = {\langle left | right \rangle}` default: left

Sets the *aligned* of the symbol defined by `mark-ans` key. The “symbol” is aligned in a box with the same dimensions of the label box defined by `labelwidth` key on the current level and separated by the value of the `labelsep` key.

6.1.3 Keys for debug and checking

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

Displays the *position* occupied by the “stored content” by `\anskey`, `anskey*`, `\item*` and `\anspic*` in the *prop list* $\{\langle store name \rangle\}$ set by `save-ans` key. This position is used by the `\getkeyans` command and by the `\ref` command if the `save-ref` key is active.

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

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

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

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

6.2 The command `\anskey`

`\anskey` \langle anskey \rangle [\langle keys \rangle] [\langle content \rangle]

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

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

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

6.2.1 Keys for `\anskey`

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

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

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

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

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

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

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

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

- | | |
|--|---|
| <ul style="list-style-type: none"> * 1. Text containing our instructions or questions. <ul style="list-style-type: none"> * <input style="border: 1px solid black; width: 150px; height: 15px;" type="text" value="first answer"/> 2. Text containing our instructions or questions. <ul style="list-style-type: none"> (a) Question. <ul style="list-style-type: none"> * <input style="border: 1px solid black; width: 150px; height: 15px;" type="text" value="second answer"/> | <ul style="list-style-type: none"> 3. Text containing our instructions or questions. <ul style="list-style-type: none"> * <input style="border: 1px solid black; width: 150px; height: 15px;" type="text" value="third answer"/> 4. Text containing our instructions or questions. <ul style="list-style-type: none"> * <input style="border: 1px solid black; width: 150px; height: 15px;" type="text" value="fourth answer"/> |
|--|---|

6.3 The environment `anskey*`

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

The environment `anskey*` takes a mandatory $\{\langle$ body content $\rangle\}$ and “stores it” in the *sequence* and *prop list* $\{\langle$ store name $\rangle\}$ set by `save-ans` key. If `save-ref` key are active and the `hyperref`[8] package is detected `\hyperlink` and `\hypertarget` will be used, otherwise the usual “label and ref” system provided by L^AT_EX will be used. By design the environment cannot be nested but full supports “verbatim material” in the body and it is assumed that each *numbered \item* or *\item** within the environment in which it is active it has a “single execution” unless *\item* or *\item** open a nested level or use the `no-store` key.

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

6.3.1 Keys for anskey*

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

`write-env = {⟨file.ext⟩}` default: *not used*

Sets the name of the `⟨external file⟩` in which the `⟨contents⟩` of the environment will be written. The `⟨file.ext⟩` will be created in the working directory, relative or absolute paths are not supported. If `⟨file.ext⟩` does not exist, it will be created or overwritten if the `overwrite` key is used.

`overwrite = {⟨true | false⟩}` default: *false*

Sets whether the `⟨file.ext⟩` generated by `write-env` from the `anskey*` environment will be rewritten.

`force-eol = {⟨true | false⟩}` default: *false*

Sets if the `end of line` for the `⟨stored content⟩` is hidden or not. This key is necessary only if the last line is the closing of some environment defined by the `fancyvrb` package as `\end{Verbatim}` or another environment that does not support a comments “%” after closing `\end{Verbatim}%`.

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

Example

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

  \item Text containing our instructions or questions.

  \begin{enumext}
    \item Question.
    \begin{anskey*}
      ⟨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. | 7. Text containing our instructions or questions. |
| [5] <input type="text" value="First answer with verbatim"/> | [7] <input type="text" value="third answer"/> |
| 6. Text containing our instructions or questions. | 8. Text containing our instructions or questions. |
| (a) Question. | [8] <input type="text" value="fourth answer"/> |
| [6] <input type="text" value="second answer"/> | |

6.4 The environments keyans and keyans*

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

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

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

```

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

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

```

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

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

```

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

```

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

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

Example

```




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

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

  \item Text containing a question and image.

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

```

- | | | | | | | | | | | | | | |
|---|---|---------------------|-----------|-----------|-----------|--|---|-----------|---|-----------|-----------|-----------|----------------------------|
| <ol style="list-style-type: none"> Text containing a question. <table border="0" style="margin-left: 20px;"> <tr> <td>A) Choice</td> <td>* B) Correct choice</td> </tr> <tr> <td>C) Choice</td> <td>D) Choice</td> </tr> <tr> <td>E) Choice</td> <td></td> </tr> </table> | A) Choice | * B) Correct choice | C) Choice | D) Choice | E) Choice | | <ol style="list-style-type: none"> Text containing a question and image. <table border="0" style="margin-left: 20px;"> <tr> <td>A) Choice</td> <td rowspan="5" style="vertical-align: middle; text-align: center;">  </td> </tr> <tr> <td>B) Choice</td> </tr> <tr> <td>C) Choice</td> </tr> <tr> <td>D) Choice</td> </tr> <tr> <td>* E) [note] Correct choice</td> </tr> </table> | A) Choice |  | B) Choice | C) Choice | D) Choice | * E) [note] Correct choice |
| A) Choice | * B) Correct choice | | | | | | | | | | | | |
| C) Choice | D) Choice | | | | | | | | | | | | |
| E) Choice | | | | | | | | | | | | | |
| A) Choice |  | | | | | | | | | | | | |
| B) Choice | | | | | | | | | | | | | |
| C) Choice | | | | | | | | | | | | | |
| D) Choice | | | | | | | | | | | | | |
| * E) [note] Correct choice | | | | | | | | | | | | | |

6.5 The environment `keyanspic`

```

keyanspic \begin{keyanspic}[<key = val>] \anspic* [<content>] {<drawing or tabular>} \end{keyanspic}

```

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

When the `keyanspic` environment is used *without keys* the $\langle labels \rangle$ 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.

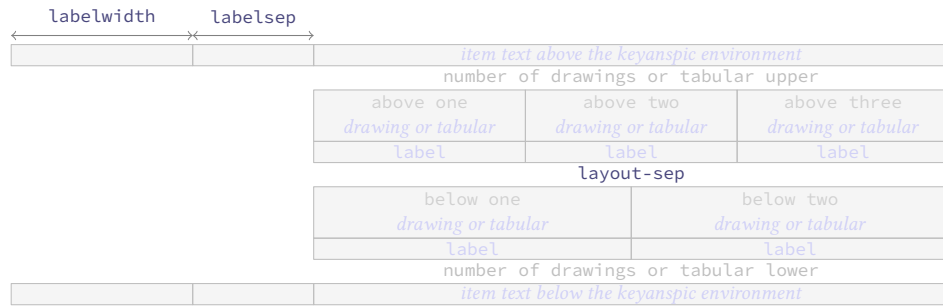


Figure 6: Representation of the `keyanspic` environment with `layout-sty={3,2}` in `enumext`.

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.

6.5.2 The command `\anspic`

`\anspic` `\anspic{<drawing or tabular>}`
`\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,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.

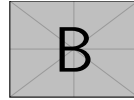
\begin{keyanspic}
\anspic{\includegraphics[scale=0.15]{example-image-a}}
\anspic{\includegraphics[scale=0.15]{example-image-b}}
\anspic{\includegraphics[scale=0.15]{example-image-a}}
\anspic{\includegraphics[scale=0.15]{example-image-a}}
\anspic*[note]{\includegraphics[scale=0.15]{example-image-a}}
\end{keyanspic}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

```

1. Question with images and labels below.



A)



B)



C)

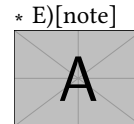
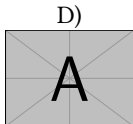
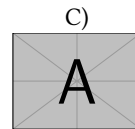
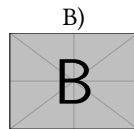
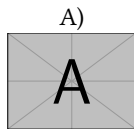


D)



* E)[note]

2. Question with images and labels above.



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*.

<code>start = {⟨integer⟩}</code>	default: 1
Sets the <i>position</i> of the <i>prop list</i> {⟨store name⟩} from which execution will start. The value must be a ⟨positive integer⟩.	
<code>stop = {⟨integer⟩}</code>	default: 0
Sets the <i>position</i> of the <i>prop list</i> {⟨store name⟩} from which execution will finish. The value must be a ⟨positive integer⟩.	
<code>before = {⟨code⟩}</code>	default: empty
Sets the {⟨code⟩} that will be executed ⟨before⟩ each {⟨content⟩} stored in <i>prop list</i> {⟨store name⟩}. The {⟨code⟩} must be passed between braces.	
<code>after = {⟨code⟩}</code>	default: empty
Sets the {⟨code⟩} that will be executed ⟨after⟩ each {⟨content⟩} stored in <i>prop list</i> {⟨store name⟩}. The {⟨code⟩} must be passed between braces.	
<code>wrapper = {⟨code #1⟩ more code}</code>	default: empty
Wraps the {⟨content⟩} stored in <i>prop list</i> {⟨store name⟩} referenced by #1. The {⟨code⟩} must be passed between braces. For example <code>\foreachkeys[wrapper={\makebox[1em][l]{#1}}]{⟨store name⟩}</code> .	

6.6.3 The command `\printkeys`

```

\printkeys {⟨store name⟩}
\printkeys [⟨keys⟩]{⟨store name⟩}
\printkeys * [⟨keys⟩]{⟨store name⟩}

```

The command `\printkeys` 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 `\printkeys*{⟨store name⟩}` and the *sequence* {⟨store name⟩} already contains any `enumext*` environment an error will be returned as we cannot nest.
- If we execute `\printkeys*{⟨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 `\printkeys{⟨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 `\printkeys` commands and `\printkeys*` 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}
  \end{enumext}

```



```

\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~~e is cool?

[2]

3. Related to Linux

(a) You use linux?

[3]

(b) Rate the following package and class

i. `xsim`

[4]

ii. `exsheets`

[5]

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

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

2. (a) Very True! *

3. (a) Yes *

(b) i. very good *

ii. obsolete *

7 Full examples

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

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

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

Example 1

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

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

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: $1 \text{ \AA} = 1 \times 10^{-10} \text{ m}$) e il fermi o femtometro ($1 \text{ fm} = 1 \times 10^{-15} \text{ m}$). 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: $1 \text{ \AA} = 1 \times 10^{-10} \text{ m}$) e il fermi o femtometro ($1 \text{ fm} = 1 \times 10^{-15} \text{ m}$). 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>

(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](#).

1

Which choice best describes what happens in the passage?

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

3

Which choice best describes what happens in the passage?

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

2

Which choice best describes what happens in the passage?

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

4

Which choice best describes what happens in the passage?

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

1. A)

2. C)

3. B)

4. D)

8 Tagged PDF examples





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

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

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

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

- The file `enumext-01.tex` contains the basic tests for the `enumext` and `enumext*` environments and the nesting between them plus the use of the `label`, `labelwidth`, `labelsep`, `ref`, `align` and `wrap-label` keys. Source file [📄](#) and `tagged` PDF [📄](#).
- The file `enumext-02.tex` contains the tests for the `enumext` and `enumext*` environments and the support for `minipage` and `multicol`s 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 $\langle keys \rangle$ 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[\langle content \rangle]` 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 $\langle labels \rangle$ 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 ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.

Final notes

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

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

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

Why list environments?

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

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

Why not random questions and other utilities

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

Why has it taken so long?

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

10 References

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

11 Change history

v1.0 2024-11-01 – First public release.

12 Index of Documentation

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

C		F	
Document class:		<code>\footnote</code>	5
<i>article</i>	2	I	
<i>book</i>	2	<code>\itemsep</code>	8
<i>exam</i>	2	K	
<i>letter</i>	2	Keys for <code>\anskey</code> provide by <code>enumext</code> :	
<i>report</i>	2	<i>break-col</i>	13
<code>\columnbreak</code>	4, 13	<i>item-join</i>	13
<code>\columnsep</code>	10	<i>item-pos*</i>	13
Commands provide by <code>enumext</code> :		<i>item-star</i>	13
<code>\anskey</code>	11–14	<i>item-sym*</i>	13
<code>\anspic</code>	11, 12, 15, 16	Keys for <code>\foreachkeyans</code> provide by <code>enumext</code> :	
<code>\foreachkeyans</code>	17	<i>after</i>	18
<code>\getkeyans</code>	12, 17	<i>before</i>	18
<code>\item*</code>	5–7, 11, 12, 14, 15	<i>sep</i>	17
<code>\item</code>	5–7, 10–12, 14, 16	<i>start</i>	17, 18
<code>\miniright</code>	11	<i>step</i>	17
<code>\printkeyans</code>	6, 12, 18	<i>stop</i>	18
<code>\setenumextmeta</code>	6	<i>wrapper</i>	18
<code>\setenumext</code>	5–7, 11, 12, 15, 18	Keys for <code>anskey*</code> provide by <code>enumext</code> :	
Counters defined by <code>enumext</code> :		<i>break-col</i>	13
<i>enumXiii</i>	4	<i>force-eol</i>	14
<i>enumXii</i>	4	<i>item-join</i>	13
<i>enumXiv</i>	4	<i>item-pos*</i>	13
<i>enumXi</i>	4	<i>item-star</i>	13
<i>enumXviii</i>	4	<i>item-sym*</i>	13
<i>enumXvii</i>	4	<i>overwrite</i>	14
<i>enumXvi</i>	4	<i>write-env</i>	14
<i>enumXv</i>	4	Keys for environments provide by <code>enumext</code> :	
E		<i>above*</i>	9
Environments provide by <code>enumext</code> :		<i>above</i>	8, 9
<code>anskey*</code>	11–14, 21	<i>after</i>	10
<code>enumext*</code>	4–11, 13–15, 18, 21, 22	<i>align</i>	7, 21–23
<code>enumext</code>	4–11, 13–16, 18, 21, 22	<i>base-fix</i>	8
<code>keyans*</code>	4–14, 22	<i>before*</i>	9, 10
<code>keyanspic</code>	4, 7, 8, 11–16, 21, 22	<i>before</i>	9
<code>keyans</code>	4–16, 22	<i>below*</i>	9
Environments:		<i>below</i>	9
<i>Verbatim</i>	14	<i>check-ans</i>	12, 13
<i>center</i>	5	<i>columns-sep</i>	4, 10, 21
<i>description</i>	5, 22	<i>columns</i>	4, 9, 10, 21
<i>enumerate</i>	1, 3, 5, 23	<i>first</i>	10
<i>figure</i>	5	<i>font</i>	7
<i>flushleft</i>	5	<i>item-pos*</i>	5, 6
<i>flushright</i>	5	<i>item-sym*</i>	5, 6
<i>itemize</i>	5, 22	<i>itemindent</i>	8–10
<i>list</i>	3, 5, 9, 23	<i>itemsep</i>	8
<i>minipage</i>	3–5, 8–11, 21, 23	<i>label-pos</i>	16
<i>multicols</i>	3, 4, 10, 21	<i>label-sep</i>	16
<i>quotation</i>	5	<i>labelsep</i>	3–7, 9, 10, 12, 21, 22
<i>quote</i>	5	<i>labelwidth</i>	3, 4, 6, 7, 9, 10, 12, 21, 22
<i>shortenumerate</i>	5	<i>labelwith</i>	5
<i>tabbing</i>	5	<i>label</i>	7, 8, 10, 15, 21–23
<i>table</i>	5	<i>labewidth</i>	9
<i>tasks</i>	5	<i>layout-sep</i>	16
<i>trivlist</i>	5	<i>layout-sty</i>	16
<i>verbatim</i>	5	<i>layout-top</i>	16
<i>verse</i>	5	<i>list-indent</i>	3, 9
		<i>list-offset</i>	3, 8, 9, 22, 23

listparindent	9, 10
mark-ans	12
mark-pos	12
mark-ref	12
mini-env	4, 9, 11, 21
mini-right*	7, 11
mini-right	7, 11, 21
mini-sep	4, 11
mode-box	7
no-store	11–13, 22
noitemsep	8
nosep	8, 22
overwrite	14
parsep	8, 10, 16
partopsep	8
ref	4, 8, 21
resume*	7, 10, 11
resume	7, 10, 11
rightmargin	9
save-ans	4, 6, 10–18, 21, 22
save-key	10–12, 18
save-ref	4, 7, 12, 13, 17, 21
save-sep	12, 21
series	7, 10, 11
show-ans	12, 22
show-length	8
show-pos	12, 17
start*	10
start	10
topsep	8, 9, 16
widest	7
wrap-ans	12
wrap-label*	7, 23
wrap-label	7, 21, 23
wrap-opt	12
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	23
enumext	1–5, 7, 16, 21–24
enumitem	3, 4, 23
fancyvrb	14
footnotehyper	5
geometry	21
graphicx	21
hyperref	4, 5, 12, 13, 21, 23
l3keys	7
l3prop	23
l3seq	23
luamml	21
multicol	1, 2, 4, 21, 23
scontents	1, 2, 14, 21
shortlst	5
tasks	5
task	6
unicode-math	21
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}[2024-11-01]
```

Now declare the `enumext` package.

```
4 \ProvidesExplPackage {enumext} {2024-11-01} {1.0} {Enumerate exercise sheets}
```

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

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

13.4 Definition of variables

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

```
\__enumext_level_int Integer variables will control the nesting levels of the environments and \anskey command.
\__enumext_level_h_int
\__enumext_anskey_level_int
\__enumext_keyans_level_int
\__enumext_keyans_level_h_int
\__enumext_keyans_pic_level_int
24 \int_new:N \__enumext_level_int
25 \int_new:N \__enumext_level_h_int
26 \int_new:N \__enumext_anskey_level_int
27 \int_new:N \__enumext_keyans_level_int
28 \int_new:N \__enumext_keyans_level_h_int
29 \int_new:N \__enumext_keyans_pic_level_int
```

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

```

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

```

Internal variables used by functions `__enumext_is_not_nested:`, `__enumext_is_on_first_level:` and `__enumext_keyans_name_and_start:` (§13.5.1).

```

30 \bool_new:N \l__enumext_starred_bool
31 \bool_new:N \g__enumext_starred_bool
32 \bool_new:N \l__enumext_starred_first_bool
33 \bool_new:N \l__enumext_standar_bool
34 \bool_new:N \g__enumext_standar_first_bool
35 \bool_new:N \l__enumext_anskey_env_bool
36 \bool_new:N \l__enumext_keyans_env_bool
37 \bool_new:N \g__enumext_start_line_tl
38 \tl_new:N \g__enumext_envir_name_tl
39 \tl_new:N \l__enumext_envir_name_tl
40 \tl_new:N \l__enumext_envir_name_tl

```

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

```

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

```

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

The initial values of these variables are set by the function `__enumext_define_counters:Nn` (§13.11) and then modified by the function `__enumext_label_style:Nnn` used by `label` key (§13.14).

```

41 \cs_set_protected:Npn \__enumext_tmp:n #1
42 {
43   \tl_new:c { l__enumext_counter_#1_tl }
44 }
45 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }

```

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

```

\c__enumext_counter_style_tl
\l__enumext_ref_key_arg_tl
\l__enumext_ref_the_count_tl
\l__enumext_the_counter_X_tl
  \l__enumext_renew_the_count_X_tl

```

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

```

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

```

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

```

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

```

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

```

57 \int_new:N \g__enumext_resume_int
58 \int_new:N \g__enumext_resume_vii_int
59 \tl_new:N \l__enumext_resume_name_tl
60 \bool_new:N \l__enumext_resume_active_bool
61 \tl_new:N \g__enumext_standar_series_tl
62 \tl_new:N \g__enumext_starred_series_tl

```

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

```

\l__enumext_current_widest_dim
\g__enumext_counter_styles_tl
\g__enumext_widest_label_tl
\l__enumext_label_width_by_box

```

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

```

63 \dim_new:N \l__enumext_current_widest_dim
64 \tl_new:N \g__enumext_counter_styles_tl
65 \tl_new:N \g__enumext_widest_label_tl
66 \box_new:N \l__enumext_label_width_by_box

```

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


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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```

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

```

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

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

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

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

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

```

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

```

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

```

\l__enumext_setkey_tmpa_tl
\l__enumext_setkey_tmpb_tl
\l__enumext_setkey_tmpa_int
\l__enumext_setkey_tmpa_seq
\l__enumext_setkey_tmpb_seq

```

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

```

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

```

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

```

\l__enumext_meta_path_tl
\l__enumext_foreach_print_seq
\l__enumext_foreach_name_prop_tl
\g__enumext_foreach_default_keys_tl

```

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

```

122 \tl_new:N \l__enumext_meta_path_tl
123 \seq_new:N \l__enumext_foreach_print_seq
124 \tl_new:N \l__enumext_foreach_name_prop_tl
125 \tl_new:N \g__enumext_foreach_default_keys_tl

```

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

```

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

```

Internal variables used by command `\printkeyans` (§13.47), `show-pos` key (§13.27), `item-sym*` key (§13.35), `save-key` key (§13.27.2) and “*storing structure*”.

```

126 \tl_new:N \l__enumext_print_keyans_starred_tl
127 \bool_new:N \l__enumext_print_keyans_star_bool
128 \str_new:N \l__enumext_mark_position_str
129 \tl_new:N \g__enumext_item_symbol_aux_tl
130 \cs_set_protected:Npn \l__enumext_tmp:n #1
131 {
132   \tl_new:c { \l__enumext_print_keyans_#1_tl }
133   \tl_new:c { \l__enumext_store_save_key_#1_tl }
134   \bool_new:c { \l__enumext_store_save_key_#1_bool }
135   \bool_new:c { \l__enumext_store_upper_level_#1_bool }
136 }
137 \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
\g__enumext_keyans_pic_parsep_skip
\l__enumext_anspic_label_box
\l__enumext_anspic_body_box
\l__enumext_anspic_label_htdp_dim
\l__enumext_anspic_body_htdp_dim

```

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

```

138 \seq_new:N \l__enumext_anspic_args_seq
139 \dim_new:N \l__enumext_anspic_mini_width_dim
140 \int_new:N \l__enumext_anspic_above_int
141 \int_new:N \l__enumext_anspic_below_int
142 \bool_new:N \l__enumext_anspic_label_above_bool
143 \str_new:N \l__enumext_anspic_mini_pos_str
144 \skip_new:N \g__enumext_keyans_pic_parsep_skip
145 \box_new:N \l__enumext_anspic_label_box
146 \box_new:N \l__enumext_anspic_body_box
147 \dim_new:N \l__enumext_anspic_label_htdp_dim
148 \dim_new:N \l__enumext_anspic_body_htdp_dim

```

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

```

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

```

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

```

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

```

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

```

\l__enumext_hyperref_bool
\l__enumext_footnotes_key_bool

```

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

```

157 \bool_new:N \l__enumext_hyperref_bool
158 \bool_new:N \l__enumext_footnotes_key_bool

```

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

```

\l__enumext_newlabel_arg_one_tl
\l__enumext_newlabel_arg_two_tl
\l__enumext_write_aux_file_tl
\l__enumext_label_copy_X_tl

```

Internal variables used by `save-ref` key (§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.

```

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

```

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

```

\g__enumext_footnote_standar_int
\g__enumext_footnote_starred_int
\g__enumext_footnote_standar_arg_seq
\g__enumext_footnote_starred_arg_seq
\g__enumext_footnote_standar_int_seq
\g__enumext_footnote_starred_int_seq

```

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

```

167 \int_new:N \g__enumext_footnote_standar_int
168 \int_new:N \g__enumext_footnote_starred_int
169 \seq_new:N \g__enumext_footnote_standar_arg_seq
170 \seq_new:N \g__enumext_footnote_starred_arg_seq
171 \seq_new:N \g__enumext_footnote_standar_int_seq
172 \seq_new:N \g__enumext_footnote_starred_int_seq

```

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

```

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

```

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

```

173 \cs_set_protected:Npn \__enumext_tmp:n #1
174 {
175   \bool_new:c { l__enumext_item_starred_#1_bool }
176   \int_new:c { l__enumext_item_column_pos_#1_int }
177   \int_new:c { g__enumext_item_count_all_#1_int }
178   \int_new:c { l__enumext_joined_item_#1_int }
179   \int_new:c { l__enumext_joined_item_aux_#1_int }
180   \int_new:c { l__enumext_tmpa_#1_int }
181   \dim_new:c { l__enumext_tmpa_#1_dim }
182   \box_new:c { l__enumext_item_text_#1_box }
183   \dim_new:c { l__enumext_joined_width_#1_dim }
184   \dim_new:c { l__enumext_item_width_#1_dim }
185   \tl_new:c { g__enumext_item_symbol_aux_#1_tl }
186   \str_new:c { l__enumext_align_label_#1_str }
187   \bool_new:c { g__enumext_minipage_active_#1_bool }
188   \box_new:c { l__enumext_miniright_code_#1_box }
189   \bool_new:c { g__enumext_minipage_center_#1_bool }
190   \dim_new:c { g__enumext_minipage_right_#1_dim }
191   \skip_new:c { g__enumext_minipage_right_#1_skip }
192 }
193 \clist_map_inline:nn { vii, viii } { \__enumext_tmp:n {#1} }

```

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

```
\c__enumext_all_envs_clist An internal clist-var variable to run with \__enumext_tmp:n.
194 \clist_const:Nn \c__enumext_all_envs_clist
195 {
196   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
197   {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
198 }
```

(End of definition for `\c__enumext_all_envs_clist`.)

13.5 Some utility functions

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

```
\seq_use:NV
199 \cs_generate_variant:Nn \keys_precompile:nnN { neN }
200 \cs_generate_variant:Nn \seq_use:Nn { NV }
```

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

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

```
201 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
202 {
203   \hook_gput_code:nnn {begindocument} {enumext} { #1 }
204 }
```

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

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

```
\__enumext_before_env:nn
205 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
206 {
207   \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
208 }
209 \cs_new_protected:Npn \__enumext_before_env:nn #1 #2
210 {
211   \hook_gput_code:nnn {env/#1/before} {enumext} {#2}
212 }
```

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

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

```
213 \cs_new:Nn \__enumext_level:
214 {
215   \int_to_roman:n { \__enumext_level_int }
216 }
```

(End of definition for `__enumext_level:.`)

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

```
217 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
218 {
219   \regex_match:nnTF { ^[\+|-]?[\d]+$ } {#1} % $
220   { \prg_return_true: }
221   { \prg_return_false: }
222 }
```

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

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

```
223 \cs_new_protected:Nn \__enumext_regex_counter_style:
224 {
225   \tl_map_inline:Nn \c__enumext_counter_style_tl
226   {
227     \regex_replace_once:nnN { \c{##1}\* }
228     { \c{##1}\cB{\u{__enumext_ref_the_count_tl}\cE} } \__enumext_ref_key_arg_tl
229   }
230 }
```

(End of definition for `__enumext_regex_counter_style:`)

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

```

231 \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
232 {
233   * ~ #2
234   \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
235   = ~ \use:c { #1_use:c } { \__enumext_#2_#3_#1 } \\
236 }

```

(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.

```

237 \cs_new_protected:Nn \__enumext_unskip_unkern:
238 {
239   \int_case:nnT { \lastnodetype }
240   {
241     { 11 }{ \unskip }
242     { 12 }{ \unkern }
243   }
244 }

```

(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:`

```

245 \cs_new_protected:Nn \__enumext_is_not_nested:
246 {
247   \str_case:en { \@currentenv }
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_standar_bool } }
254       { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
255       {
256         \bool_gset_true:N \g__enumext_standar_bool
257       }
258     }
259     {enumext*}
260     {
261       \tl_set:Nn \l__enumext_envir_name_tl { enumext* }
262       \bool_lazy_and:nnT
263       { \bool_not_p:n { \g__enumext_starred_bool } }
264       { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
265       {
266         \bool_gset_true:N \g__enumext_starred_bool
267       }
268     }
269   }
270 }

```

The function `__enumext_is_on_first_level:` will set the variables `\l__enumext_standar_first_bool` (§13.26.1), `\l__enumext_starred_first_bool` (§13.26.1) and `\l__enumext_anskey_env_bool` (§13.31) to “true” only if the environment is not nested and we are in the “first level” of it . We will also save the *start line number* of each environment in the variable `\g__enumext_start_line_tl` and the *name* of each environment in the variable `\g__enumext_envir_name_tl` to use in messages related to the `check-ans` key and `.log` file.

```

271 \cs_new_protected:Nn \__enumext_is_on_first_level:
272 {
273   \bool_lazy_all:nT
274   {
275     { \bool_if_p:N \g__enumext_standar_bool }
276     { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
277     { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
278   }

```

```

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

```

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

`__enumext_keyans_name_and_start:`

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

```

304 \cs_new_protected:Nn \__enumext_keyans_name_and_start:
305 {
306   \str_case:en { \@currentenv }
307   {
308     {keyans}
309     {
310       \tl_set:Nn \l__enumext_envir_name_tl { keyans }
311       \tl_set:Ne \l__enumext_check_start_line_env_tl
312         {
313           in ~ 'keyans' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
314         }
315     }
316     {keyans*}
317     {
318       \tl_set:Nn \l__enumext_envir_name_tl { keyans* }
319       \tl_set:Ne \l__enumext_check_start_line_env_tl
320         {
321           in ~ 'keyans*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
322         }
323     }
324     {keyanspic}
325     {
326       \tl_set:Nn \l__enumext_envir_name_tl { keyanspic }
327       \tl_set:Ne \l__enumext_check_start_line_env_tl
328         {
329           in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
330         }
331     }
332   }
333 }

```

(End of definition for `__enumext_keyans_name_and_start:`)

13.5.2 Utilities for log and terminal

`__enumext_reset_global_vars:`
`__enumext_reset_global_int:`
`__enumext_reset_global_bool:`
`__enumext_reset_global_tl:`

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

```

334 \cs_new_protected:Nn \__enumext_reset_global_vars:
335 {
336   \__enumext_reset_global_int:
337   \__enumext_reset_global_bool:

```



```

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

```

(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.

```

358 \cs_new_protected:Nn \__enumext_log_global_vars:
359   {
360   \msg_log:nneeee { enumext } { prop-seq-int-hook }
361   { \g__enumext_store_name_tl }
362   { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
363   { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
364   { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
365   }

```

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.

```

366 \cs_new_protected:Nn \__enumext_log_answer_vars:
367   {
368   \msg_log:nneee { enumext } { item-answer-hook }
369   { \int_use:N \g__enumext_item_number_int }
370   { \int_use:N \g__enumext_item_anskey_int }
371   { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int } }
372   }

```

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

13.6 Copying list and minipage environments

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

```

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

```

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

```

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

```

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

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

`__enumext_start_list:nn` 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.

```

\__enumext_stop_list:
\__enumext_item_std:w
\__enumext_minipage:w
\__enumext_endminipage:
373 \__enumext_at_begin_document:n
374   {
375   \cs_new_eq:NN \__enumext_start_list:nn \list

```

```

376     \cs_new_eq:NN \__enumext_stop_list: \endlist
377     \NewCommandCopy \__enumext_item_std:w \item
378   }

```

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

```

379 \__enumext_at_begin_document:n
380 {
381   \cs_new_eq:NN \__enumext_minipage:w \minipage
382   \cs_new_eq:NN \__enumext_endminipage: \endminipage
383 }

```

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

13.7 Compatibility with hyperref and footnotehyper

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

```

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

```

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

```

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

```

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`.

```

399 \bool_if:NT \l__enumext_footnotes_key_bool
400 {
401   \IfPackageLoadedTF { footnotehyper }
402   {
403     \msg_info:nnn { enumext } { package-load } { footnotehyper }
404   }
405   {
406     \bool_set_false:N \l__enumext_footnotes_key_bool
407   }
408 }

```

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

```

409 \bool_if:NTF \l__enumext_hyprerref_bool
410 {
411   \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
412   \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
413 }
414 {
415   \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
416   \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
417 }
418 }

```

(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.

```

419 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
420 {
421   \protected@write \@auxout { }
422   {
423     \token_to_str:N \newlabel {#1}
424     {
425       {#2}
426       \bool_if:NT \l__enumext_hyperref_bool
427       { { \thepage } {#2} {#1} }
428       { }
429     }
430   }
431   \__enumext_hypertarget:nn {#1} { }
432   \__enumext_phantomsection:
433 }

```

(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.

```

434 \cs_new_protected:Nn \__enumext_footnotetext:nn
435 {
436   \footnotetext[#1]{#2}
437 }
438 \cs_new_protected:Nn \__enumext_renew_footnote:
439 {
440   \RenewDocumentCommand \footnote { o +m }
441   {
442     \tl_if_novalue:nTF {##1}
443     {
444       \stepcounter{footnote}
445       \int_gset_eq:Nc \g__enumext_footnote_standar_int { c@footnote }
446     }
447     {
448       \int_gset:Nn \g__enumext_footnote_standar_int { ##1 }
449     }
450     \footnotemark [ \g__enumext_footnote_standar_int ]
451     \seq_gput_right:Nn \g__enumext_footnote_standar_arg_seq { ##2 }
452     \seq_gput_right:NV
453     \g__enumext_footnote_standar_int_seq \g__enumext_footnote_standar_int
454   }
455 }
456 \cs_new_protected:Nn \__enumext_print_footnote:
457 {
458   \seq_if_empty:NF \g__enumext_footnote_standar_int_seq
459   {
460     \seq_map_pairwise_function:NNN
461     \g__enumext_footnote_standar_int_seq
462     \g__enumext_footnote_standar_arg_seq
463     \__enumext_footnotetext:nn
464   }
465   \seq_gclear:N \g__enumext_footnote_standar_arg_seq
466   \seq_gclear:N \g__enumext_footnote_standar_int_seq
467 }

```

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.

```

468 \cs_new_protected:Nn \__enumext_renew_footnote_mini:
469 {
470   \RenewDocumentCommand \footnote { o +m }
471   {
472     \tl_if_novalue:nTF {##1}
473     {

```

```

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

```

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

`__enumext_renew_footnote_standar:` 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.

```

\__enumext_renew_footnote_standar:
\__enumext_print_footnote_standar:
\__enumext_renew_footnote_starred:
\__enumext_print_footnote_starred:
498 \cs_new_protected:Nn \__enumext_renew_footnote_standar:
499 {
500   \bool_if:NT \g__enumext_standar_bool
501   {
502     \IfDocumentMetadataTF
503     {
504       \__enumext_renew_footnote:
505     }
506     {
507       \bool_if:NF \l__enumext_footnotes_key_bool
508       {
509         \__enumext_renew_footnote:
510       }
511     }
512   }
513 }
514 \cs_new_protected:Nn \__enumext_print_footnote_standar:
515 {
516   \bool_if:NT \g__enumext_standar_bool
517   {
518     \IfDocumentMetadataTF
519     {
520       \__enumext_print_footnote:
521     }
522     {
523       \bool_if:NF \l__enumext_footnotes_key_bool
524       {
525         \__enumext_print_footnote:
526       }
527     }
528   }
529 }

```

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.

```

530 \cs_new_protected:Nn \__enumext_renew_footnote_starred:
531 {
532   \IfDocumentMetadataTF
533   {
534     \__enumext_renew_footnote_mini:
535   }

```

```

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

```

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

```

556 \__enumext_after_env:nn { enumext* }
557 {
558     \__enumext_print_footnote_starred:
559 }
560 \__enumext_after_env:nn { keyans* }
561 {
562     \__enumext_print_footnote_starred:
563 }

```

(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)

```

564 \cs_new_protected:Nn \__enumext_internal_mini_page:
565 {
566     \int_compare:nNt { \l__enumext_level_int } = { 0 }
567     {
568         \DeclareDocumentEnvironment{__enumext_mini_page}{ m }
569         {
570             \__enumext_renew_footnote_standar:
571             \__enumext_minipage:w [ t ] { #1 }
572             \legacy_if_gset_false:n { @minipage }
573             \skip_vertical:N \c_zero_skip
574         }
575         {
576             \skip_vertical:N \c_zero_skip
577             \__enumext_endminipage:
578             \__enumext_print_footnote_standar:
579         }
580     }
581 }

```

(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.

```

582 \dim_zero_new:N \itemwidth

```

13.11 Definition of counters

```

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

```

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

#1: A token list `\l__enumext_counter_X_tl` for “store” the counter’s name.

#2: The counter’s name.

```

583 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
584 {
585   \cs_if_exist:cTF { c@ #2 }
586   { \msg_fatal:nnn { enumext } { counters } { #2 } }
587   {
588     \tl_set:Nn #1 { #2 }
589     \newcounter { #2 }
590   }
591 }

```

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.

```

592 \__enumext_define_counters:Nn \l__enumext_counter_i_tl { enumXi }
593 \__enumext_define_counters:Nn \l__enumext_counter_ii_tl { enumXii }
594 \__enumext_define_counters:Nn \l__enumext_counter_iii_tl { enumXiii }
595 \__enumext_define_counters:Nn \l__enumext_counter_iv_tl { enumXiv }
596 \__enumext_define_counters:Nn \l__enumext_counter_v_tl { enumXv }
597 \__enumext_define_counters:Nn \l__enumext_counter_vi_tl { enumXvi }
598 \__enumext_define_counters:Nn \l__enumext_counter_vii_tl { enumXvii }
599 \__enumext_define_counters:Nn \l__enumext_counter_viii_tl { enumXviii }

```

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

13.12 Definition of labels

This part of the code is inspired by the `enumitem` package. The idea is to be able to access the counters using `\arabic*`, `\Alph*`, `\alph*`, `\Roman*` and `\roman*` to use them in the `label` key.

```

\__enumext_register_counter_style:Nn

```

These `<counters>` will be used as default `<labels>` if the `label` key is not used for the different levels of the `enumext`, `enumext*`, `keyans` and `keyans*` environments, so it is necessary to get a default value for `labelwidth` from these `<labels>` at the same time.

```

600 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
601 {
602   \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
603   \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
604 }
605 \__enumext_register_counter_style:Nn \arabic { 0 }
606 \__enumext_register_counter_style:Nn \Alph { M }
607 \__enumext_register_counter_style:Nn \alph { m }
608 \__enumext_register_counter_style:Nn \Roman { VIII }
609 \__enumext_register_counter_style:Nn \roman { viii }

```

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

```

\__enumext_label_width_by_box:Nn
\__enumext_label_width_by_box:cv

```

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

```

610 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
611 {
612   \hbox_set:Nn \l__enumext_label_width_by_box {#2}
613   \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
614 }
615 \cs_generate_variant:Nn \__enumext_label_width_by_box:Nn { cv }

```

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

```

\__enumext_label_style:Nnn
\__enumext_label_style:cvn

```

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

```

616 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
617 {
618   \tl_clear_new:N #1

```



```

619 \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
620 \tl_gset_eq:NN \g__enumext_widest_label_tl #1
621 \tl_map_inline:Nn \g__enumext_counter_styles_tl
622 {
623   \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
624   \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
625   { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
626 }
627 \__enumext_label_width_by_box:Nn \__enumext_current_widest_dim
628 { \tl_use:N \g__enumext_widest_label_tl }
629 \tl_set_eq:cN { the #2 } #1
630 }
631 \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.

```

632 \cs_set_protected:Npn \__enumext_tmp:n #1
633 {
634   \keys_define:nn { enumext / #1 }
635   {
636     mode-box .bool_set:N = \__enumext_mode_box_bool,
637     mode-box .initial:n = false,
638     mode-box .value_forbidden:n = true,
639   }
640 }
641 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }

```

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

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

```

642 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
643 {
644   \keys_define:nn { enumext / #1 }
645   {
646     font .tl_set:c = { \__enumext_label_font_style_#2_tl },
647     font .value_required:n = true,
648     labelsep .dim_set:c = { \__enumext_labelsep_#2_dim },
649     labelsep .initial:n = {0.3333em},
650     labelsep .value_required:n = true,
651     labelwidth .dim_set:c = { \__enumext_labelwidth_#2_dim },
652     labelwidth .value_required:n = true,
653     wrap-label .cs_set_protected:cp = { \__enumext_wrapper_label_#2:n } ##1,
654     wrap-label .initial:n = {##1},
655     wrap-label .value_required:n = true,
656     wrap-label* .code:n = {
657       \bool_set_true:c { \__enumext_wrap_label_opt_#2_bool }
658       \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
659     },
660     wrap-label* .value_required:n = true,
661   }
662 }
663 \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 `__enumext_align_label_pos_X_str`.

```

664 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
665 {
666   \keys_define:nn { enumext / #1 }
667   {
668     align .choice:,
669     align / left .code:n =
670     {

```

```

671         \tl_clear:c { l__enumext_label_fill_left_#2_tl }
672         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
673         \str_set:cn { l__enumext_align_label_pos_#2_str } { l }
674     },
675     align / right .code:n =
676     {
677         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
678         \tl_clear:c { l__enumext_label_fill_right_#2_tl }
679         \str_set:cn { l__enumext_align_label_pos_#2_str } { r }
680     },
681     align / center .code:n =
682     {
683         \tl_set:cn { l__enumext_label_fill_left_#2_tl } { \hfill }
684         \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
685         \str_set:cn { l__enumext_align_label_pos_#2_str } { c }
686     },
687     align / unknown .code:n =
688         \msg_error:nnee { enumext } { unknown-choice }
689         { align } { left, ~ right, ~ center } { \exp_not:n {##1} },
690     align .initial:n = left,
691     align .value_required:n = true,
692 }
693 }
694 \clist_map_inline:nn
695 {
696     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
697 }
698 { \__enumext_tmp:nn #1 }

699 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
700 {
701     \keys_define:nn { enumext / #1 }
702     {
703         align .choice:,
704         align / left .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
705         align / right .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
706         align / center .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
707         align / unknown .code:n =
708             \msg_error:nnee { enumext } { unknown-choice }
709             { align } { left, ~ right, ~ center } { \exp_not:n {##1} },
710         align .initial:n = left,
711         align .value_required:n = true,
712     }
713 }
714 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }

```

(End of definition for align.)

13.14 Setting label and ref keys

The implementation of the keys `label` and `ref` are part of the core of the package `enumext`, here the default values for $\langle label \rangle$, the value of the variables `\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 $\langle labels \rangle$ of the *four levels* of `enumext` environment, along with the default value for `ref` `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
715 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
716 {
717     \keys_define:nn { enumext / #1 }
718     {
719         label .code:n = {
720             \__enumext_label_style:cnv { l__enumext_label_#2_tl }
721             { l__enumext_counter_#2_tl } {##1}
722             \dim_set_eq:cN { l__enumext_labelwidth_#2_dim }
723             \l__enumext_current_widest_dim
724         },
725         label .initial:n = #3,
726         label .value_required:n = true,
727         ref .code:n = \__enumext_standar_ref:n {##1},
728         ref .value_required:n = true,
729     }

```

```

730 }
731 \__enumext_tmp:nnn { level-1 } { i } { \arabic*.}
732 \__enumext_tmp:nnn { level-2 } { ii } { (\alph*) }
733 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
734 \__enumext_tmp:nnn { level-4 } { iv } { \Alph*. }

```

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

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

```

735 \cs_new_protected:Npn \__enumext_standar_ref:n #1
736 {
737   \tl_set:Nn \__enumext_ref_key_arg_tl {#1}
738   \tl_if_empty:NTF \__enumext_ref_key_arg_tl
739   {
740     \msg_error:nnn { enumext } { key-ref-empty } { enumext }
741   }
742   {
743     \tl_set_eq:Nc
744     \__enumext_ref_the_count_tl { \__enumext_counter_ \__enumext_level: _tl }
745     \__enumext_regex_counter_style:
746     \tl_set_eq:Nc
747     \__enumext_ref_the_count_tl { \__enumext_the_counter_ \__enumext_level: _tl }
748     \tl_put_right:ce { \__enumext_renew_the_count_ \__enumext_level: _tl }
749     {
750       \exp_not:N \renewcommand { \exp_not:V \__enumext_ref_the_count_tl } { \exp_not:V \__
751     }
752   }
753 }

```

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

```

754 \cs_new_protected:Nn \__enumext_standar_ref:
755 {
756   \tl_if_empty:cF { \__enumext_renew_the_count_ \__enumext_level: _tl }
757   {
758     \tl_use:c { \__enumext_renew_the_count_ \__enumext_level: _tl }
759   }
760 }

```

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

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

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

```

\__enumext_label_vii_tl 761 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
\__enumext_label_viii_tl 762 {
763   \keys_define:nn { enumext / #1 }
764   {
765     label .code:n = {
766       \__enumext_label_style:cvn { \__enumext_label_#2_tl }
767       { \__enumext_counter_#2_tl } {##1}
768       \dim_set_eq:cN { \__enumext_labelwidth_#2_dim }
769       \__enumext_current_widest_dim
770     },
771     label .initial:n = #3,
772     label .value_required:n = true,
773     ref .code:n = \__enumext_starred_ref:n {##1},
774     ref .value_required:n = true,
775   }
776 }
777 \__enumext_tmp:nnn { enumext* } { vii } { \arabic*.}
778 \__enumext_tmp:nnn { keyans* } { viii } { \Alph*.}

```

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

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

```

779 \cs_new_protected:Npn \__enumext_starred_ref:n #1
780 {
781   \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
782   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
783   {
784     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
785     {
786       \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
787     }
788     {
789       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
790       \__enumext_regex_counter_style:
791       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
792       \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
793       {
794         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V
795       }
796     }
797   }
798   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
799   {
800     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
801     {
802       \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
803     }
804     {
805       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
806       \__enumext_regex_counter_style:
807       \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
808       \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
809       {
810         \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V
811       }
812     }
813   }
814 }

```

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.

```

815 \cs_new_protected:Nn \__enumext_starred_ref:
816 {
817   \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
818   {
819     \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
820     {
821       \tl_use:N \l__enumext_renew_the_count_vii_tl
822     }
823   }
824   \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
825   {
826     \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
827     {
828       \tl_use:N \l__enumext_renew_the_count_viii_tl
829     }
830   }
831 }

```

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

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

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

```

\l__enumext_label_v_tl
\l__enumext_label_vi_tl
832 \keys_define:nn { enumext / keyans }
833 {
834   label .code:n = {
835     \__enumext_label_style:cvn { \l__enumext_label_v_tl }
836     { \l__enumext_counter_v_tl } {#1}
837     \dim_set_eq:cN { \l__enumext_labelwidth_v_dim }
838     \l__enumext_current_widest_dim
839     \__enumext_label_style:cvn { \l__enumext_label_vi_tl }

```

```

840         { l__enumext_counter_vi_tl } {#1}
841         \dim_set_eq:cN { l__enumext_labelwidth_v_dim }
842         \l__enumext_current_widest_dim
843     },
844     label .initial:n = \Alph*),
845     label .value_required:n = true,
846     ref .code:n = \__enumext_keyans_ref:n {#1},
847     ref .value_required:n = true,
848 }

```

(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:n
\__enumext_keyans_ref:n
849 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
850 {
851     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
852     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
853     {
854         \msg_error:nnn { enumext } { key-ref-empty } { keyans }
855     }
856     {
857         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
858         \__enumext_regex_counter_style:
859         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
860         \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
861         {
862             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V \l__
863         }
864     }
865 }

```

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

```

866 \cs_new_protected:Npn \__enumext_keyans_ref:
867 {
868     \tl_if_empty:NF \l__enumext_renew_the_count_v_tl
869     {
870         \tl_use:N \l__enumext_renew_the_count_v_tl
871     }
872 }

```

(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.

```

873 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
874 {
875     \__enumext_if_is_int:nTF { #3 }
876     {
877         \int_set:Nn #2 {#3}
878     }
879     {
880         \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
881         { \int_set:Nn #2 { \int_from_alph:n {#3} } }
882         \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
883         { \int_set:Nn #2 { \int_from_roman:n {#3} } }
884     }
885 }
886 \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.

```
887 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
888 {
889   \__enumext_if_is_int:nTF {#4}
890   {
891     \setcounter{enumX#1} { #4 }
892   }
893   {
894     \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
895     { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
896     \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
897     { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
898   }
899   \__enumext_label_width_by_box:cv
900   { \l__enumext_labelwidth_#1_dim } { \l__enumext_label_#1_tl }
901 }
902 \cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }
```

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

start Now define and set `start*`, `start` and `widest` keys for `enumext`, `enumext*`, `keyans` and `keyans*` environ-
start* ments.

```
widest
903 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
904 {
905   \keys_define:nn { enumext / #1 }
906   {
907     start* .code:n = {
908       \__enumext_start_from:ccn
909       { \l__enumext_label_#2_tl }
910       { \l__enumext_start_#2_int } {##1}
911     },
912     start* .value_required:n = true,
913     start .code:n = {
914       \__enumext_start_from:cce
915       { \l__enumext_label_#2_tl }
916       { \l__enumext_start_#2_int } { \int_eval:n {##1} }
917     },
918     start .initial:n = 1,
919     start .value_required:n = true,
920     widest .code:n = {
921       \__enumext_widest_from:nccn {#2}
922       { \l__enumext_label_#2_tl }
923       { \l__enumext_labelwidth_#2_dim } {##1}
924     },
925     widest .value_required:n = true,
926   }
927 }
928 \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

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

```
parsep
noitemsep
nosep
929 \cs_set_protected:Npn \__enumext_tmp:nnnnn #1 #2 #3 #4 #5 #6
930 {
931   \keys_define:nn { enumext / #1 }
932   {
933     topsep .skip_set:c = { \l__enumext_topsep_#2_skip },
934     topsep .initial:n = {#3},
935     topsep .value_required:n = true,
936     partopsep .skip_set:c = { \l__enumext_partopsep_#2_skip },
937     partopsep .initial:n = {#4},
938     partopsep .value_required:n = true,
```



```

939     parsep     .skip_set:c = { l__enumext_parsep_#2_skip },
940     parsep     .initial:n  = {#5},
941     parsep     .value_required:n = true,
942     itemsep    .skip_set:c = { l__enumext_itemsep_#2_skip },
943     itemsep    .initial:n  = {#6},
944     itemsep    .value_required:n = true,
945     noitemsep  .meta:n      = { itemsep = 0pt, parsep = 0pt },
946     noitemsep  .value_forbidden:n = true,
947     nosepp    .meta:n      = {
948         itemsep = 0pt, parsep= 0pt,
949         topsep = 0pt, partopsep = 0pt,
950     },
951     nosepp    .value_forbidden:n = true,
952 }
953 }

```

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

```

954 \__enumext_tmp:nnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
955   { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
956   { 4.0pt plus 2.0pt minus 1.0pt }
957 \__enumext_tmp:nnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
958   { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
959   { 2.0pt plus 1.0pt minus 1.0pt }
960 \__enumext_tmp:nnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
961   { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
962 \__enumext_tmp:nnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
963   { 1.0pt minus 1.0pt } { 0pt } { 2.0pt plus 1.0pt minus 1.0pt }
964 \__enumext_tmp:nnnnn { keyans } { v } { 4.0pt plus 2.0pt minus 1.0pt }
965   { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
966   { 2.0pt plus 1.0pt minus 1.0pt }
967 \__enumext_tmp:nnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
968   { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
969   { 4.0pt plus 2.0pt minus 1.0pt }
970 \__enumext_tmp:nnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
971   { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
972   { 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.

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

```

base-fix \__enumext_nested_base_line_fix:
973 \keys_define:nn { enumext / level-1 }
974   {
975     base-fix .bool_set:N = \l__enumext_base_line_fix_bool,
976     base-fix .initial:n  = false,
977     base-fix .value_forbidden:n = true,
978   }

```

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.

```

979 \cs_new_protected:Nn \__enumext_nested_base_line_fix:
980   {
981     \bool_lazy_all:nT
982       {
983         { \bool_if_p:N \l__enumext_starred_first_bool }
984         { \bool_if_p:N \l__enumext_base_line_fix_bool }
985         { \bool_not_p:n { \l__enumext_print_keyans_star_bool } }
986       }
987     {
988       \mode_leave_vertical:

```

```

989     \vspace { -\dim_eval:n { \baselineskip + \parsep } }
990   }

```

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`.

```

991   \bool_lazy_and:nnT
992     { \bool_if_p:N \l__enumext_starred_first_bool }
993     { \bool_if_p:N \l__enumext_print_keyans_star_bool }
994     {
995       \mode_leave_vertical:
996       \skip_vertical:n { -\baselineskip }
997       \skip_vertical:N \c_zero_skip
998     }

```

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

```

999   \keys_set:nn { enumext / level-1 }
1000   {
1001     topsep = 0pt, above = 0pt, above* = 0pt,
1002   }
1003   \bool_set_false:N \l__enumext_base_line_fix_bool
1004 }

```

(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.

```

1005 \cs_set_protected:Npn \l__enumext_tmp:nn #1 #2
1006 {
1007   \keys_define:nn { enumext / #1 }
1008   {
1009     itemindent .dim_set:c = { \l__enumext_fake_item_indent_#2_dim },
1010     itemindent .value_required:n = true,
1011     rightmargin .dim_set:c = { \l__enumext_rightmargin_#2_dim },
1012     rightmargin .value_required:n = true,
1013     listparindent .dim_set:c = { \l__enumext_listparindent_#2_dim },
1014     listparindent .value_required:n = true,
1015     list-offset .dim_set:c = { \l__enumext_listoffset_#2_dim },
1016     list-offset .value_required:n = true,
1017     list-indent .code:n =
1018       \bool_set_true:c { \l__enumext_leftmargin_tmp_#2_bool }
1019       \dim_set:cn { \l__enumext_leftmargin_tmp_#2_dim } {##1},
1020     list-indent .value_required:n = true,
1021   }
1022 }
1023 \clist_map_inline:nn
1024 {
1025   {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
1026 }
1027 { \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.

```

1028 \cs_set_protected:Npn \l__enumext_tmp:nn #1 #2
1029 {
1030   \keys_define:nn { enumext / #1 }
1031   {
1032     itemindent .dim_set:c = { \l__enumext_fake_item_indent_#2_dim },
1033     itemindent .value_required:n = true,
1034     rightmargin .dim_set:c = { \l__enumext_rightmargin_#2_dim },
1035     rightmargin .value_required:n = true,
1036     listparindent .dim_set:c = { \l__enumext_listparindent_#2_dim },
1037     listparindent .value_required:n = true,
1038     list-offset .dim_set:c = { \l__enumext_listoffset_#2_dim },
1039     list-offset .value_required:n = true,
1040     list-indent .meta:n = { list-offset = ##1 },
1041     list-indent .value_required:n = true,
1042   }

```

```

1043 }
1044 \clist_map_inline:nn
1045 {
1046   {enumext*}{vii}, {keyans*}{viii}
1047 }
1048 { \__enumext_tmp:nn #1 }

```

13.18.1 Functions for setting the fake itemindent

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

```

1049 \cs_set_protected:Nn \__enumext_fake_item_indent:
1050 {
1051   \dim_compare:nNnT
1052     { \dim_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _dim } }
1053     >
1054     { \c_zero_dim }
1055     {
1056       \tl_set:ce { \l__enumext_fake_item_indent_ \__enumext_level: _tl }
1057       {
1058         \exp_not:N \mode_leave_vertical:
1059         \exp_not:n { \skip_horizontal:n }
1060         { \dim_use:c { \l__enumext_fake_item_indent_ \__enumext_level: _dim } }
1061         \exp_not:N \ignorespaces
1062       }
1063     }
1064 }
1065 \cs_set_protected:Nn \__enumext_keyans_fake_item_indent:
1066 {
1067   \dim_compare:nNnT
1068     { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
1069     {
1070       \tl_set:Ne \l__enumext_fake_item_indent_v_tl
1071       {
1072         \exp_not:N \mode_leave_vertical:
1073         \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
1074         \exp_not:N \ignorespaces
1075       }
1076     }
1077 }
1078 \cs_set_protected:Nn \__enumext_fake_item_indent_vii:
1079 {
1080   \dim_compare:nNnT
1081     { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
1082     {
1083       \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
1084       {
1085         \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
1086         \exp_not:N \ignorespaces
1087       }
1088     }
1089 }
1090 \cs_set_protected:Nn \__enumext_fake_item_indent_viii:
1091 {
1092   \dim_compare:nNnT
1093     { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
1094     {
1095       \tl_set:Ne \l__enumext_fake_item_indent_viii_tl
1096       {
1097         \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
1098         \exp_not:N \ignorespaces
1099       }
1100     }
1101 }

```

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

13.19 Setting show-length key

`show-length` Define and set `show-length` key for `enumext`, `enumext*`, `keyans` and `keyans*` environments. The function sets the boolean variable `\l__enumext_show_length_X_bool` used in the definition of all environments to

“true” and calls the function `__enumext_show_length:nnn` which prints all the values of the “vertical” and “horizontal” parameters calculated and used.

```

1102 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1103 {
1104   \keys_define:nn { enumext / #1 }
1105   {
1106     show-length .bool_set:c = { l__enumext_show_length_#2_bool },
1107     show-length .initial:n = false,
1108   }
1109 }
1110 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

(End of definition for show-length.)

13.20 Setting before, after and first keys

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

```

1111 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1112 {
1113   \keys_define:nn { enumext / #1 }
1114   {
1115     before .tl_set:c = { l__enumext_before_no_starred_key_#2_tl },
1116     before .value_required:n = true,
1117     before* .tl_set:c = { l__enumext_before_starred_key_#2_tl },
1118     before* .value_required:n = true,
1119     after .tl_set:c = { l__enumext_after_stop_list_#2_tl },
1120     after .value_required:n = true,
1121     first .tl_set:c = { l__enumext_after_list_args_#2_tl },
1122     first .value_required:n = true,
1123   }
1124 }
1125 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }

```

(End of definition for before and others.)

13.20.1 Functions for before, after and first keys in enumext

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

```

1126 \cs_new_protected:Nn \__enumext_before_args_exec:
1127 {
1128   \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
1129 }

```

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}`

```

1130 \cs_new_protected:Nn \__enumext_before_keys_exec:
1131 {
1132   \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
1133 }

```

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

```

1134 \cs_new_protected:Nn \__enumext_after_stop_list:
1135 {
1136   \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
1137 }

```

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.`

```

1138 \cs_new_protected:Nn \__enumext_after_args_exec:
1139 {
1140   \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
1141 }

```

(End of definition for __enumext_before_args_exec: and others.)

13.20.2 Functions for before, after and first keys in keyans

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

```

\__enumext_before_keys_exec_v: 1142 \cs_new_protected:Nn \__enumext_before_args_exec_v:
\__enumext_after_stop_list_v: 1143 {
\__enumext_after_args_exec_v: 1144   \tl_use:N \l__enumext_before_starred_key_v_tl
                               1145 }
                               1146 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
                               1147 {
                               1148   \tl_use:N \l__enumext_before_no_starred_key_v_tl
                               1149 }
                               1150 \cs_new_protected:Nn \__enumext_after_stop_list_v:
                               1151 {
                               1152   \tl_use:N \l__enumext_after_stop_list_v_tl
                               1153 }
                               1154 \cs_new_protected:Nn \__enumext_after_args_exec_v:
                               1155 {
                               1156   \tl_use:N \l__enumext_after_list_args_v_tl
                               1157 }

```

(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*

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

```

\__enumext_before_keys_exec_vii 1158 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
\__enumext_after_stop_list_vii: 1159 {
\__enumext_after_args_exec_vii: 1160   \tl_use:N \l__enumext_before_starred_key_vii_tl
                               1161 }
                               1162 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
                               1163 {
                               1164   \tl_use:N \l__enumext_before_starred_key_viii_tl
                               1165 }
                               1166 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
                               1167 {
                               1168   \tl_use:N \l__enumext_before_no_starred_key_vii_tl
                               1169 }
                               1170 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
                               1171 {
                               1172   \tl_use:N \l__enumext_before_no_starred_key_viii_tl
                               1173 }
                               1174 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
                               1175 {
                               1176   \tl_use:N \l__enumext_after_stop_list_vii_tl
                               1177 }
                               1178 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
                               1179 {
                               1180   \tl_use:N \l__enumext_after_stop_list_viii_tl
                               1181 }
                               1182 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
                               1183 {
                               1184   \tl_use:N \l__enumext_after_list_args_vii_tl
                               1185 }
                               1186 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
                               1187 {
                               1188   \tl_use:N \l__enumext_after_list_args_viii_tl
                               1189 }

```

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

13.21 Setting keys for multicol and minipage

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

```

1190 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1191 {
1192   \keys_define:nn { enumext / #1 }
1193   {
1194     mini-env .dim_set:c = { l__enumext_minipage_right_#2_dim },
1195     mini-env .value_required:n = true,
1196     mini-sep .dim_set:c = { l__enumext_minipage_hsep_#2_dim },

```

```

1197     mini-sep .initial:n = 0.3333em,
1198     mini-sep .value_required:n = true,
1199     columns-sep .dim_set:c = { l__enumext_columns_sep_#2_dim },
1200     columns-sep .value_required:n = true,
1201     columns .int_set:c = { l__enumext_columns_#2_int },
1202     columns .initial:n = 1,
1203     columns .value_required:n = true,
1204   }
1205 }
1206 \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.

```

1207 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1208 {
1209   \keys_define:nn { enumext / #1 }
1210   {
1211     mini-right .tl_gset:c = { g__enumext_miniright_code_#2_tl },
1212     mini-right .value_required:n = true,
1213     mini-right* .code:n = {
1214       \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
1215       \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1216     },
1217     mini-right* .value_required:n = true,
1218   }
1219 }
1220 \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.

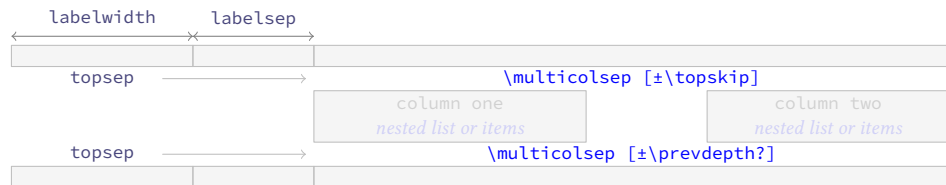


Figure 7: Representation of the vertical space in `multicols` for a nested level.

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*.

```

1221 \cs_new_protected:Nn \__enumext_multi_set_vskip:
1222 {
1223   \skip_set:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1224   {
1225     \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1226   }
1227   \skip_set:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1228   {
1229     \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1230   }

```



```

1231   \__enumext_add_pre_parsep:
1232   }

```

(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*.

```

1233 \cs_new_protected:Nn \__enumext_add_pre_parsep:
1234 {
1235   \int_case:nn { \l__enumext_level_int }
1236   {
1237     { 2 }{
1238       \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1239       {
1240         \skip_add:Nn \l__enumext_multicols_above_ii_skip
1241         {
1242           \l__enumext_parsep_i_skip
1243         }
1244       }
1245     }
1246     { 3 }{
1247       \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1248       {
1249         \skip_add:Nn \l__enumext_multicols_above_iii_skip
1250         {
1251           \l__enumext_parsep_ii_skip
1252         }
1253       }
1254     }
1255     { 4 }{
1256       \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1257       {
1258         \skip_add:Nn \l__enumext_multicols_above_iv_skip
1259         {
1260           \l__enumext_parsep_iii_skip
1261         }
1262       }
1263     }
1264   }
1265 }

```

(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*.

```

1266 \cs_new_protected:Nn \__enumext_multi_addvspace:
1267 {
1268   \__enumext_multi_set_vskip:
1269   \mode_if_vertical:T
1270   {
1271     \skip_add:cn { l__enumext_multicols_above_ \l__enumext_level: _skip }
1272     {
1273       \skip_use:c { l__enumext_partopsep_ \l__enumext_level: _skip }
1274     }
1275     \skip_add:cn { l__enumext_multicols_below_ \l__enumext_level: _skip }
1276     {
1277       \skip_use:c { l__enumext_partopsep_ \l__enumext_level: _skip }
1278     }
1279   }
1280   \par\nopagebreak
1281   \addvspace{ \skip_use:c { l__enumext_multicols_above_ \l__enumext_level: _skip } }
1282 }

```

(End of definition for `__enumext_multi_addvspace:`)

13.22.2 Adjustment of vertical spaces for multicol in keyans

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

```

1283 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1284 {
1285   \skip_set:Nn \l__enumext_multicol_above_v_skip
1286     {
1287       \l__enumext_topsep_v_skip
1288     }
1289   \skip_set:Nn \l__enumext_multicol_below_v_skip
1290     {
1291       \l__enumext_topsep_v_skip
1292     }
1293 }
1294 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1295 {
1296   \__enumext_keyans_multi_set_vskip:
1297   \mode_if_vertical:T
1298   {
1299     \skip_add:Nn \l__enumext_multicol_above_v_skip
1300       {
1301         \skip_use:N \l__enumext_partopsep_v_skip
1302       }
1303     \skip_add:Nn \l__enumext_multicol_below_v_skip
1304       {
1305         \skip_use:N \l__enumext_partopsep_v_skip
1306       }
1307   }
1308   \par\nopagebreak
1309   \addvspace{ \l__enumext_multicol_above_v_skip }
1310 }

```

(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 it nested. The setting of the values was more “trial and error” (aprox to `\strutbox`), using the help of the `lua-visual-debug`[14] package, again my attempts to find the correct values using `\showoutput` and `\showboxdepth` absolutely failed.

13.23.1 Adjustment of vertical spaces for minipage in enumext

`__enumext_minipage_set_skip:` 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`.

```

1311 \cs_new_protected:Nn \__enumext_minipage_set_skip:
1312 {
1313   \skip_set:Nn \l__enumext_minipage_right_skip

```

```

1314     {
1315       \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1316     }
1317   \mode_if_vertical:T
1318     {
1319       \skip_add:Nn \l__enumext_minipage_right_skip
1320       {
1321         \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1322       }
1323     }
1324   \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:`.

```

1325   \skip_set_eq:cN
1326     { l__enumext_multicols_above_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1327   \skip_set_eq:cN
1328     { l__enumext_multicols_below_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1329   \__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`.

```

1330   \int_compare:nNnT
1331     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1332     {
1333       \skip_zero:N \topskip
1334       \skip_set_eq:Nc \multicolsep { l__enumext_multicols_above_ \__enumext_level: _skip }
1335     }
1336 }

```

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*.

```

1337 \cs_new_protected:Nn \__enumext_minipage_add_space:
1338 {
1339   \__enumext_minipage_set_skip:
1340   \__enumext_unskip_unkern:
1341   \mode_if_vertical:TF
1342     {
1343       \nopagebreak\nointerlineskip
1344     }
1345     {
1346       \par\nopagebreak\nointerlineskip
1347       \skip_zero:c { l__enumext_partopsep_ \__enumext_level: _skip }
1348     }
1349   \int_compare:nNnTF
1350     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1351     {
1352       \addvspace{ 0.445\box_ht:N \strutbox }
1353     }
1354     {
1355       \addvspace{ 0.250\box_ht:N \strutbox }
1356     }
1357 }

```

(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.

```

1358 \cs_new_protected:Nn \__enumext_pre_itemsep_skip:
1359 {
1360   \int_case:nn { \l__enumext_level_int }
1361     {
1362       { 2 }{
1363         \skip_if_eq:nnTF
1364           { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1365           {
1366             \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1367             \skip_set:Nn \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }

```

```

1368     }
1369     {
1370     \dim_compare:nNnT
1371     { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1372     {
1373     \skip_sub:Nn
1374     \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1375     \skip_sub:Nn
1376     \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1377     \skip_add:Nn
1378     \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1379     \skip_add:Nn
1380     \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1381     }
1382     \dim_compare:nNnT
1383     { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1384     {
1385     \skip_set:Nn \l__enumext_minipage_temp_skip
1386     {
1387     \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1388     }
1389     \skip_sub:Nn
1390     \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1391     \skip_sub:Nn
1392     \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1393     \skip_add:Nn
1394     \l__enumext_minipage_after_skip
1395     { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1396     \skip_add:Nn
1397     \l__enumext_multicols_below_ii_skip
1398     { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1399     }
1400     }
1401     }
1402     { 3 }{
1403     \skip_if_eq:nnTF
1404     { \l__enumext_itemsep_ii_skip } { \c_zero_skip }
1405     {
1406     \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1407     \skip_set:Nn \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1408     }
1409     {
1410     \dim_compare:nNnT
1411     { \l__enumext_itemsep_ii_skip } < { \l__enumext_minipage_after_skip }
1412     {
1413     \skip_sub:Nn
1414     \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1415     \skip_sub:Nn
1416     \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1417     \skip_add:Nn
1418     \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1419     \skip_add:Nn
1420     \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1421     }
1422     \dim_compare:nNnT
1423     { \l__enumext_itemsep_ii_skip } > { \l__enumext_minipage_after_skip }
1424     {
1425     \skip_set:Nn \l__enumext_minipage_temp_skip
1426     {
1427     \l__enumext_itemsep_ii_skip - \l__enumext_minipage_after_skip
1428     }
1429     \skip_sub:Nn
1430     \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1431     \skip_sub:Nn
1432     \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1433     \skip_add:Nn
1434     \l__enumext_minipage_after_skip
1435     { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1436     \skip_add:Nn
1437     \l__enumext_multicols_below_iii_skip
1438     { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }

```

```

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

```

(End of definition for `\l__enumext_pre_itemsep_skip`.)

13.23.2 Adjustment of vertical spaces for minipage in keyans

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

`\l__enumext_keyans_minipage_add_space`:
`\l__enumext_keyans_pre_itemsep_skip`:

```

1483 \cs_new_protected:Nn \l__enumext_keyans_minipage_set_skip:
1484 {
1485   \skip_zero:N \l__enumext_minipage_after_skip
1486   \skip_zero:N \l__enumext_minipage_left_skip
1487   \skip_zero:N \l__enumext_minipage_right_skip
1488   \skip_set:Nn \l__enumext_minipage_right_skip
1489   {
1490     \l__enumext_topsep_v_skip
1491   }
1492   \mode_if_vertical:T
1493   {
1494     \skip_add:Nn \l__enumext_minipage_right_skip
1495     {
1496       \l__enumext_partopsep_v_skip
1497     }
1498   }
1499   \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip
1500   \skip_set_eq:NN \l__enumext_multicols_above_v_skip \l__enumext_minipage_right_skip
1501   \skip_set_eq:NN \l__enumext_multicols_below_v_skip \l__enumext_minipage_right_skip
1502   \l__enumext_keyans_pre_itemsep_skip:

```

```

1503 \int_compare:nNt { \l__enumext_columns_v_int } > { 1 }
1504 {
1505     \skip_zero:N \topskip
1506     \skip_set_eq:NN \multicolsep \l__enumext_minipage_right_skip
1507 }
1508 }
1509 \cs_new_protected:Nn \l__enumext_keyans_minipage_add_space:
1510 {
1511     \l__enumext_keyans_minipage_set_skip:
1512     \l__enumext_unskip_unkern:
1513     \mode_if_vertical:TF
1514     {
1515         \nopagebreak\nointerlineskip
1516     }
1517     {
1518         \par\nopagebreak\nointerlineskip
1519         \skip_zero:N \l__enumext_partopsep_v_skip
1520     }
1521     \int_compare:nNtTF { \l__enumext_columns_v_int } > { 1 }
1522     {
1523         \addvspace{ 0.445\box_ht:N \strutbox }
1524     }
1525     {
1526         \addvspace{ 0.250\box_ht:N \strutbox }
1527     }
1528 }
1529 \cs_new_protected:Nn \l__enumext_keyans_pre_itemsep_skip:
1530 {
1531     \skip_if_eq:nnTF
1532     { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1533     {
1534         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1535         \skip_set:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1536     }
1537     {
1538         \dim_compare:nNt
1539         { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1540         {
1541             \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1542             \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1543             \skip_add:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1544             \skip_add:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1545         }
1546         \dim_compare:nNt
1547         { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1548         {
1549             \skip_set:Nn \l__enumext_minipage_temp_skip
1550             {
1551                 \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1552             }
1553             \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1554             \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1555             \skip_add:Nn \l__enumext_minipage_after_skip
1556             { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1557             \skip_add:Nn \l__enumext_multicols_below_v_skip
1558             { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1559         }
1560     }
1561 }

```

(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:` 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*`.

```

1562 \cs_new_protected:Nn \l__enumext_mini_set_vskip_vii:
1563 {
1564     \skip_zero_new:N \l__enumext_minipage_left_skip
1565     \skip_gzero_new:N \g__enumext_minipage_right_skip

```



```

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

```

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

`__enumext_mini_addvspace_vii:` 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.

`__enumext_mini_addvspace_viii:`

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.

```

1619 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1620 {
1621   \__enumext_mini_set_vskip_vii:
1622   \par\nopagebreak
1623   \addvspace { \l__enumext_minipage_left_skip }
1624 }
1625 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1626 {
1627   \__enumext_mini_set_vskip_viii:
1628   \par\nopagebreak

```

```

1629   \addvspace { \l__enumext_minipage_left_skip }
1630   }

```

(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.

```

1631 \NewDocumentCommand \miniright { s }
1632 {
1633   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1634   {
1635     \msg_error:nnn { enumext } { wrong-miniright-place }
1636   }
1637   % outside
1638   \bool_lazy_and:nnT
1639   { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
1640   { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
1641   {
1642     \msg_error:nnn { enumext } { wrong-miniright-place }
1643   }
1644   % starred env
1645   \bool_lazy_and:nnT
1646   { \bool_if_p:N \g__enumext_starred_bool }
1647   { \bool_not_p:n { \l__enumext_standar_bool } }
1648   {
1649     \msg_error:nnn { enumext } { wrong-miniright-starred }
1650   }
1651   % exec
1652   \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
1653   {
1654     \__enumext_keyans_mini_right_cmd:n {#1}
1655   }
1656   { \__enumext_mini_right_cmd:n {#1} }
1657 }

```

(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.

```

1658 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1659 {
1660   \dim_compare:nNnTF
1661   { \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1662   {
1663     \__enumext_multicols_stop:
1664     \int_compare:nNnT
1665     { \int_use:c { \l__enumext_columns_ \__enumext_level: _int } } = { 1 }
1666     {
1667       \par\addvspace{ \l__enumext_minipage_after_skip }
1668     }
1669     \end__enumext_mini_page
1670     \hfill
1671     \__enumext_mini_page{ \dim_use:c { \l__enumext_minipage_right_ \__enumext_level: _dim } }
1672     \par\nointerlineskip
1673     \addvspace { \l__enumext_minipage_right_skip }
1674     \bool_if:nF {#1}
1675     {
1676       \centering

```

```

1677     }
1678     \int_gzero:N \g__enumext_minipage_stat_int
1679   }
1680   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1681 % paranoia
1682 \RenewDocumentCommand \miniright { s }
1683 {
1684   \msg_error:nn { enumext } { many-miniright-used }
1685 }
1686 }

```

(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.

```

1687 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1688 {
1689   \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1690   {
1691     \__enumext_keyans_multicols_stop:
1692     \int_compare:nNnT { \l__enumext_columns_v_int } = { 1 }
1693     {
1694       \par\addvspace{ \l__enumext_minipage_after_skip }
1695     }
1696     \end__enumext_mini_page
1697     \hfill
1698     \__enumext_mini_page{ \l__enumext_minipage_right_v_dim }
1699     \par\nointerlineskip
1700     \addvspace { \l__enumext_minipage_right_skip }
1701     \bool_if:nF {#1}
1702     {
1703       \centering
1704     }
1705     \int_gzero:N \g__enumext_minipage_stat_int
1706   }
1707   { \msg_error:nnn { enumext } { wrong-miniright-use } }
1708 % paranoia
1709 \RenewDocumentCommand \miniright { s }
1710 {
1711   \msg_error:nn { enumext } { many-miniright-used }
1712 }
1713 }

```

(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* 1714 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
below 1715 {
below* 1716   \keys_define:nn { enumext / #1 }
1717   {
1718     above .skip_set:c = { \l__enumext_vspace_above_#2_skip },
1719     above .value_required:n = true,
1720     above* .code:n = \bool_set_true:c { \l__enumext_vspace_a_star_#2_bool }
1721             \keys_set:nn { enumext / #1 } { above = {##1} },
1722     above* .value_required:n = true,
1723     below .skip_set:c = { \l__enumext_vspace_below_#2_skip },
1724     below .value_required:n = true,
1725     below* .code:n = \bool_set_true:c { \l__enumext_vspace_b_star_#2_bool }
1726             \keys_set:nn { enumext / #1 } { below = {##1} },
1727     below* .value_required:n = true,
1728   }
1729 }
1730 \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.

```

1731 \cs_new_protected:Nn \__enumext_vspace_above:
1732 {
1733   \skip_if_eq:nnF
1734     { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1735     {
1736       \bool_if:cTF { \__enumext_vspace_a_star_ \__enumext_level: _bool }
1737         {
1738           \vspace*{ \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1739         }
1740         {
1741           \vspace { \skip_use:c { \__enumext_vspace_above_ \__enumext_level: _skip } }
1742         }
1743     }
1744 }

```

(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.

```

1745 \cs_new_protected:Nn \__enumext_vspace_below:
1746 {
1747   \skip_if_eq:nnF
1748     { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1749     {
1750       \bool_if:cTF { \__enumext_vspace_b_star_ \__enumext_level: _bool }
1751         {
1752           \vspace*{ \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1753         }
1754         {
1755           \vspace { \skip_use:c { \__enumext_vspace_below_ \__enumext_level: _skip } }
1756         }
1757     }
1758 }

```

(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.

```

1759 \cs_new_protected:Nn \__enumext_vspace_above_v:
1760 {
1761   \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1762   {
1763     \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1764       {
1765         \vspace*{ \l__enumext_vspace_above_v_skip }
1766       }
1767       { \vspace { \l__enumext_vspace_above_v_skip } }
1768   }
1769 }

```

(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.

```

1770 \cs_new_protected:Nn \__enumext_vspace_below_v:
1771 {
1772   \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1773   {
1774     \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1775       {
1776         \vspace*{ \l__enumext_vspace_below_v_skip }
1777       }
1778       { \vspace { \l__enumext_vspace_below_v_skip } }
1779   }
1780 }

```

(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.

```

1781 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1782 {
1783   \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1784   {
1785     \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1786     {
1787       \vspace*{ \l__enumext_vspace_above_vii_skip }
1788     }
1789     { \vspace { \l__enumext_vspace_above_vii_skip } }
1790   }
1791 }
1792 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1793 {
1794   \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1795   {
1796     \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1797     {
1798       \vspace*{ \l__enumext_vspace_above_viii_skip }
1799     }
1800     { \vspace { \l__enumext_vspace_above_viii_skip } }
1801   }
1802 }

```

(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.

`__enumext_vspace_below_vii:`

```

1803 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1804 {
1805   \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1806   {
1807     \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1808     {
1809       \vspace*{ \l__enumext_vspace_below_vii_skip }
1810     }
1811     { \vspace { \l__enumext_vspace_below_vii_skip } }
1812   }
1813 }
1814 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1815 {
1816   \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1817   {
1818     \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1819     {
1820       \vspace*{ \l__enumext_vspace_below_viii_skip }
1821     }
1822     { \vspace { \l__enumext_vspace_below_viii_skip } }
1823   }
1824 }

```

(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 `<keys>` passed to the *optional argument* of the “*first level*” of the environments `enumext` and `enumext*`, but, discarding some specific `<keys>`. This implementation is adapted directly from the code provided by Jonathan P. Spratte (@Skillmon) in [chat-Tex-SX](#)

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

```

1825 \cs_set_protected:Npn \__enumext_tmp:n #1
1826 {
1827   \keys_define:nn { enumext / #1 }
1828   {
1829     series .str_set:N = \l__enumext_series_str,
1830     series .value_required:n = true,
1831     resume .code:n = \__enumext_resume_series:n {##1},
1832     resume* .code:n = \__enumext_resume_starred:,

```

```

1833     resume* .value_forbidden:n = true,
1834   }
1835 }
1836 \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

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

```

\__enumext_filter_series:n
  \__enumext_filter_series_key:n
  \__enumext_filter_series_pair:nn
1837 \cs_new:Npn \__enumext_filter_series:n #1
1838 {
1839   \use:e
1840   {
1841     \keyval_parse:NNn
1842     \__enumext_filter_series_key:n
1843     \__enumext_filter_series_pair:nn {#1}
1844   }
1845 }

```

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

```

1846 \cs_new:Npn \__enumext_filter_series_key:n #1
1847 {
1848   \str_case:nnF {#1}
1849   {
1850     { resume } {} { resume* } {} { base-fix } {}
1851   }
1852   { , { \exp_not:n {#1} } }
1853 }

```

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

```

1854 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1855 {
1856   \str_case:nnF {#1}
1857   {
1858     { series } {} { resume } {} { start } {}
1859     { start* } {} { save-ans } {} { save-key } {}
1860   }
1861   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1862 }

```

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

`__enumext_parse_series:n` The function `__enumext_parse_series:n` will be responsible for storing the filtered $\langle keys \rangle$ in the global variable `\g__enumext_series_<series name>_tl` along with the creation of the integer variable `\g__enumext_series_<series name>_int` when the key is passed as an argument; otherwise, it will check the state of the boolean variable `__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 `__enumext_resume_active_bool` is set to true by the function `__enumext_resume_counter:n` which is used by the keys `resume` and `resume*`, in this case we must Make sure it is set to false so that it does not overwrite the default filtered $\langle keys \rangle$. This function is passed to the function `__enumext_parse_keys:n` in the `enumext` environment definition (§13.39) and to the function `__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§13.44).

```

1863 \cs_new_protected:Npn \__enumext_parse_series:n #1
1864 {
1865   \str_if_empty:NTF \__enumext_series_str
1866   {
1867     \bool_if:NF \__enumext_resume_active_bool
1868     {
1869       \__enumext_resume_last:n {#1}
1870     }
1871   }
1872   {
1873     \tl_gclear_new:c { g__enumext_series_ \__enumext_series_str_tl }
1874     \tl_gset:ce { g__enumext_series_ \__enumext_series_str_tl }
1875     { \__enumext_filter_series:n {#1} }
1876     \int_if_exist:cF { g__enumext_series_ \__enumext_series_str_int }
1877     {

```



```

1878         \int_new:c { g__enumext_series_ \l__enumext_series_str _int }
1879     }
1880 }
1881 }

```

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.

```

1882 \cs_new_protected:Npn \__enumext_resume_last:n #1
1883 {
1884     \bool_if:NT \l__enumext_standar_first_bool
1885     {
1886         \tl_gclear:N \g__enumext_standar_series_tl
1887         \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1888     }
1889     \bool_if:NT \l__enumext_starred_first_bool
1890     {
1891         \tl_gclear:N \g__enumext_starred_series_tl
1892         \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1893     }
1894 }

```

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

13.25.2 Internal function to save counter value

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

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

```

1895 \cs_new_protected:Npn \__enumext_resume_save_counter:
1896 {
1897     \bool_if:NT \g__enumext_standar_bool
1898     {
1899         \tl_if_empty:NF \l__enumext_series_str
1900         {
1901             \int_gset_eq:cN
1902             { g__enumext_series_ \l__enumext_series_str _int } \value{enumXi}
1903         }
1904         \tl_if_empty:NTF \l__enumext_resume_name_tl
1905         {
1906             \str_if_empty:NT \l__enumext_series_str
1907             {
1908                 \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1909             }
1910         }
1911         {
1912             \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1913             {
1914                 \int_gset_eq:cN
1915                 { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXi}
1916             }
1917         }
1918         \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1919         {
1920             \int_gset_eq:cN
1921             { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1922         }
1923     }
1924     \bool_if:NT \g__enumext_starred_bool
1925     {
1926         \tl_if_empty:NF \l__enumext_series_str
1927         {
1928             \int_gset_eq:cN
1929             { g__enumext_series_ \l__enumext_series_str _int } \value{enumXvii}

```

```

1930     }
1931     \tl_if_empty:NTF \l__enumext_resume_name_tl
1932     {
1933         \str_if_empty:NT \l__enumext_series_str
1934         {
1935             \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1936         }
1937     }
1938     {
1939         \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1940         {
1941             \int_gset_eq:cN
1942             { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXvii}
1943         }
1944     }
1945     \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1946     {
1947         \int_gset_eq:cN
1948         { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1949     }
1950 }
1951 }

```

(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.

```

1952 \cs_new_protected:Npn \__enumext_resume_series:n #1
1953 {
1954     \tl_if_empty:nTF {#1}
1955     {
1956         \__enumext_resume_counter:n { }
1957     }
1958     {
1959         \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1960         {
1961             \__enumext_resume_counter:n {#1}
1962             \bool_if:NT \g__enumext_standar_bool
1963             {
1964                 \keys_set:nv { enumext / level-1 }
1965                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1966             }
1967             \bool_if:NT \g__enumext_starred_bool
1968             {
1969                 \keys_set:nv { enumext / enumext* }
1970                 { g__enumext_series_ \tl_to_str:n {#1} _tl }
1971             }
1972         }
1973         {
1974             \bool_if:NT \g__enumext_standar_bool
1975             {
1976                 \msg_error:nnn { enumext } { unknown-series } {#1}
1977             }
1978             \bool_if:NT \g__enumext_starred_bool
1979             {
1980                 \msg_error:nnn { enumext } { unknown-series } {#1}
1981             }
1982         }
1983     }
1984 }

```

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

`__enumext_resume_counter:n`

`__enumext_resume_counter:`

`__enumext_resume_counter_series:`

`__enumext_resume_counter_save_ans:`

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

contain the $\{\langle series\ name\rangle\}$. If the variable $\backslash\l_enumext_series_name_tl$ is empty, that is, we are passing the key *resume without value*, we will execute the function $\backslash__enumext_resume_counter$: otherwise, when we pass $resume=\{\langle series\ name\rangle\}$ we will execute the function $\backslash__enumext_resume_counter_series$: finally we will execute the function $\backslash__enumext_resume_counter_save_ans$: which is associated with the key *save-ans*.

```

1985 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1986 {
1987   \bool_set_true:N \l__enumext_resume_active_bool
1988   \tl_set:Nn \l__enumext_resume_name_tl {#1}
1989   \tl_if_empty:NTF \l__enumext_resume_name_tl
1990     {
1991       \__enumext_resume_counter:
1992     }
1993     {
1994       \__enumext_resume_counter_series:
1995     }
1996   \__enumext_resume_counter_save_ans:
1997 }

```

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

```

1998 \cs_new_protected:Nn \__enumext_resume_counter:
1999 {
2000   \bool_if:NT \g__enumext_standar_bool
2001     {
2002       \int_gincr:N \g__enumext_resume_int
2003       \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
2004     }
2005   \bool_if:NT \g__enumext_starred_bool
2006     {
2007       \int_gincr:N \g__enumext_resume_vii_int
2008       \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
2009     }
2010 }

```

The function $\backslash__enumext_resume_counter_series$: 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.

```

2011 \cs_new_protected:Nn \__enumext_resume_counter_series:
2012 {
2013   \bool_if:NT \g__enumext_standar_bool
2014     {
2015       \int_set:Nn \l__enumext_start_i_int
2016         {
2017           \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
2018         }
2019     }
2020   \bool_if:NT \g__enumext_starred_bool
2021     {
2022       \int_set:Nn \l__enumext_start_vii_int
2023         {
2024           \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
2025         }
2026     }
2027 }

```

The function $\backslash__enumext_resume_counter_save_ans$: 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.

```

2028 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
2029 {
2030   \bool_lazy_and:nnT
2031     { \bool_if_p:N \l__enumext_standar_first_bool }
2032     { \bool_if_p:N \l__enumext_store_active_bool }
2033     {
2034       \int_set:Nn \l__enumext_start_i_int
2035         {
2036           \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
2037         }
2038     }
2039   \bool_lazy_and:nnT

```

```

2040     { \bool_if_p:N \__enumext_starred_first_bool }
2041     { \bool_if_p:N \__enumext_store_active_bool }
2042     {
2043       \int_set:Nn \__enumext_start_vii_int
2044       {
2045         \int_use:c { g__enumext_resume_ \__enumext_store_name_tl _int } + 1
2046       }
2047     }
2048   }

```

(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.

```

2049 \cs_new_protected:Nn \__enumext_resume_starred:
2050 {
2051   \bool_if:NT \g__enumext_standar_bool
2052   {
2053     \tl_if_empty:NF \g__enumext_standar_series_tl
2054     {
2055       \__enumext_resume_counter:n { }
2056       \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
2057     }
2058   }
2059   \bool_if:NT \g__enumext_starred_bool
2060   {
2061     \tl_if_empty:NF \g__enumext_starred_series_tl
2062     {
2063       \__enumext_resume_counter:n { }
2064       \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
2065     }
2066   }
2067 }

```

(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*`.

```

2068 \cs_set_protected:Npn \__enumext_tmp:n #1
2069 {
2070   \keys_define:nn { enumext / #1 }
2071   {
2072     save-ans .code:n = \__enumext_storing_set:n {##1},
2073     save-ans .value_required:n = true,
2074   }
2075 }
2076 \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:`.

```

2077 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
2078 {
2079   \msg_term:nnVV { enumext } { save-ans-log }
2080   \g__enumext_envir_name_tl \__enumext_store_name_tl
2081 }
2082 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
2083 {

```

```

2084     \msg_term:nnV { enumext } { save-ans-log-hook }
2085     \g__enumext_envir_name_tl \g__enumext_store_name_tl
2086 }

```

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

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

```

2087 \cs_new_protected:Npn \__enumext_storing_set:n #1
2088 {
2089     \tl_set:Ne \l__enumext_store_name_tl {#1}
2090     \tl_if_empty:NTF \l__enumext_store_name_tl
2091     {
2092         \bool_lazy_or:nnT
2093         { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
2094         {
2095             \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
2096         }
2097     }
2098     {
2099         \bool_lazy_or:nnT
2100         { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
2101         {
2102             \__enumext_start_save_ans_msg:
2103             \__enumext_storing_exec:
2104         }
2105     }
2106 }

```

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

```

2107 \cs_new_protected:Nn \__enumext_storing_exec:
2108 {
2109     \bool_set_true:N \l__enumext_store_active_bool
2110     \bool_set_true:N \l__enumext_check_answers_bool
2111     \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
2112     \__enumext_anskey_env_make:V \l__enumext_store_name_tl

```

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

```

2113     \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
2114     {
2115         \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
2116         \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
2117     }
2118     \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
2119     {
2120         \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
2121         \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
2122     }
2123     \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
2124     {
2125         \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
2126         \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
2127     }
2128 }

```

(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 2129 \cs_set_protected:Npn \__enumext_tmp:n #1
no-store   2130 {
            2131   \keys_define:nn { enumext / #1 }
            2132   {
            2133     check-ans .bool_set:N = \__enumext_check_ans_key_bool,
            2134     check-ans .initial:n = false,
            2135     check-ans .value_required:n = true,
            2136     no-store .code:n = {
            2137               \bool_set_false:N \__enumext_check_answers_bool
            2138               \bool_set_false:N \__enumext_check_ans_key_bool
            2139             },
            2140     no-store .value_forbidden:n = true,
            2141   }
            2142 }
2143 \clist_map_inline:nn
2144 {
2145   level-1, level-2, level-3, level-4, enumext*
2146 }
2147 { \__enumext_tmp:n {#1} }
```

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

13.26.5 Set-up check answer mechanism

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

```

2148 \cs_new_protected:Nn \__enumext_check_ans_active:
2149 {
2150   \tl_if_empty:NF \l__enumext_store_name_tl
2151   {
2152     \bool_if:NT \l__enumext_check_answers_bool
2153     {
2154       \__enumext_check_ans_level:
2155     }
2156   }
2157 }
```

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”.

```

2158 \cs_new_protected:Nn \__enumext_check_ans_level:
2159 {
2160   \int_case:nn { \l__enumext_level_int }
2161   {
2162     { 1 }{
2163       \bool_lazy_all:nT
2164       {
2165         { \bool_if_p:N \g__enumext_starred_bool }
2166         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
2167       }
2168       {
2169         \int_gdecr:N \g__enumext_item_number_int
2170         \bool_set_false:N \l__enumext_item_number_bool
2171       }
2172     }
2173     { 2 }{
2174       \int_gdecr:N \g__enumext_item_number_int
2175       \bool_set_false:N \l__enumext_item_number_bool
2176     }
2177     { 3 }{
2178       \int_gdecr:N \g__enumext_item_number_int
2179       \bool_set_false:N \l__enumext_item_number_bool
2180     }
2181     { 4 }{
2182       \int_gdecr:N \g__enumext_item_number_int
2183       \bool_set_false:N \l__enumext_item_number_bool
2184     }
2185   }

```

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.

```

2186   \int_case:nn { \l__enumext_level_h_int }
2187   {
2188     { 1 }{
2189       \bool_lazy_all:nT
2190       {
2191         { \bool_if_p:N \g__enumext_standar_bool }
2192         { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
2193       }
2194       {
2195         \int_gdecr:N \g__enumext_item_number_int
2196         \bool_set_false:N \l__enumext_item_number_bool
2197       }
2198     }
2199   }
2200 }

```

(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.

```

2201 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
2202 {
2203   \bool_lazy_and:nnT
2204   { \bool_if_p:N \l__enumext_check_ans_key_bool }
2205   { \bool_if_p:N \g__enumext_standar_bool }
2206   {
2207     \bool_gset_true:N \g__enumext_check_ans_key_bool
2208   }
2209   \bool_lazy_and:nnT
2210   { \bool_if_p:N \l__enumext_check_ans_key_bool }
2211   { \bool_if_p:N \g__enumext_starred_bool }
2212   {
2213     \bool_gset_true:N \g__enumext_check_ans_key_bool
2214   }
2215 }

```


(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:`.

```
2216 \cs_new_protected:Nn \__enumext_item_answer_diff:
2217 {
2218   \int_gset:Nn \g__enumext_item_answer_diff_int
2219   {
2220     \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
2221   }
2222 }
```

(End of definition for `__enumext_item_answer_diff:`)

`__enumext_check_ans_show:` The function `__enumext_check_ans_show:` will be executed within the function `__enumext_execute_after_env:` when the key `check-ans` is active, that is, when `\g__enumext_check_ans_key_bool` is “*true*” and will return the appropriate message according to the value of `\g__enumext_item_answer_diff_int` set by the function `__enumext_item_answer_diff:`

```
2223 \cs_new_protected:Nn \__enumext_check_ans_show:
2224 {
2225   \int_case:nn { \g__enumext_item_answer_diff_int }
2226   {
2227     { -1 } { \__enumext_check_ans_msg_less: }
2228     { 0 } { \__enumext_check_ans_msg_same_ok: }
2229     { 1 } { \__enumext_check_ans_msg_greater: }
2230   }
2231 }
2232 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
2233 {
2234   \msg_warning:nnee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2235   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2236 }
2237 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
2238 {
2239   \msg_term:nnee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2240   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2241 }
2242 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
2243 {
2244   \msg_warning:nnee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2245   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2246 }
```

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

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

```
2247 \cs_new_protected:Nn \__enumext_check_ans_log:
2248 {
2249   \int_case:nn { \g__enumext_item_answer_diff_int }
2250   {
2251     { -1 } { \__enumext_check_ans_log_msg_less: }
2252     { 0 } { \__enumext_check_ans_log_msg_same_ok: }
2253     { 1 } { \__enumext_check_ans_log_msg_greater: }
2254   }
2255 }
2256 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
2257 {
2258   \msg_log:nnee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2259   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2260 }
2261 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
2262 {
2263   \msg_log:nnee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2264   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2265 }
```

```

2266 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
2267 {
2268   \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2269   { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2270 }

```

(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.

```

2271 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
2272 {
2273   \int_compare:nNnT
2274     { \g__enumext_check_starred_cmd_int } = { 0 }
2275     {
2276       \msg_warning:nnnV
2277         { enumext } { missing-starred } { #1 } \l__enumext_check_start_line_env_tl
2278     }
2279   \int_compare:nNnT
2280     { \g__enumext_check_starred_cmd_int } > { 1 }
2281     {
2282       \msg_warning:nnnV
2283         { enumext } { many-starred } { #1 } \l__enumext_check_start_line_env_tl
2284     }
2285   \int_gzero:N \g__enumext_check_starred_cmd_int
2286   \tl_clear:N \l__enumext_check_start_line_env_tl
2287 }

```

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

13.27 Keys and functions associated with storage

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

We add the keys `wrap-ans`, `wrap-opt`, `save-sep`, `mark-ans`, `mark-pos`, `show-ans`, `show-pos`, `mark-ref` and `save-ref` related to the “*storage system*” and internal mechanism of “*label and ref*” only at the *first level* of `enumext` and `enumext*`.

```

2288 \cs_set_protected:Npn \__enumext_tmp:n #1
2289 {
2290   \keys_define:nn { enumext / #1 }
2291   {
2292     wrap-ans .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
2293     wrap-ans .initial:n =
2294       {
2295         \fbox{\parbox[t]{\dimeval{\itemwidth -2\fboxsep -2\fboxrule}}{##1}}
2296       },
2297     wrap-ans .value_required:n = true,
2298     wrap-opt .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
2299     wrap-opt .initial:n = [##1],
2300     wrap-opt .value_required:n = true,
2301     save-sep .tl_set:N = \l__enumext_store_keyans_item_opt_sep_tl,
2302     save-sep .initial:n = {, ~ },
2303     save-sep .value_required:n = true,
2304     mark-ans .tl_set:N = \l__enumext_mark_answer_sym_tl,
2305     mark-ans .initial:n = \textasteriskcentered,
2306     mark-ans .value_required:n = true,
2307     mark-pos .choice:,
2308     mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2309     mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2310     mark-pos / unknown .code:n =
2311       \msg_error:nneee { enumext } { unknown-choice }
2312       { mark-pos } { left, ~ right } { \exp_not:n {##1} },
2313     mark-pos .initial:n = right,
2314     mark-pos .value_required:n = true,
2315     show-ans .bool_set:N = \l__enumext_show_answer_bool,
2316     show-ans .initial:n = false,
2317     show-ans .value_required:n = true,
2318     show-pos .bool_set:N = \l__enumext_show_position_bool,
2319     show-pos .initial:n = false,
2320     show-pos .value_required:n = true,
2321     mark-ref .tl_set:N = \l__enumext_mark_ref_sym_tl,

```

```

2322     mark-ref .initial:n = \textreferencemark,
2323     mark-ref .value_required:n = true,
2324     save-ref .bool_set:N = \l__enumext_store_ref_key_bool,
2325     save-ref .initial:n = false,
2326     save-ref .value_required:n = true,
2327   }
2328 }
2329 \clist_map_inline:nn { level-1, enumext* } { \l__enumext_tmp:n {#1} }

```

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

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

```

show-ans 2330 \cs_set_protected:Npn \l__enumext_tmp:n #1
show-pos 2331 {
2332   \keys_define:nn { enumext / #1 }
2333   {
2334     mark-pos .choice:,
2335     mark-pos / left .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2336     mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2337     mark-pos .initial:n = right,
2338     mark-pos .value_required:n = true,
2339     show-ans .bool_set:N = \l__enumext_show_answer_bool,
2340     show-ans .initial:n = false,
2341     show-ans .value_required:n = true,
2342     show-pos .bool_set:N = \l__enumext_show_position_bool,
2343     show-pos .initial:n = false,
2344     show-pos .value_required:n = true,
2345   }
2346 }
2347 \clist_map_inline:nn { keyans, keyans* } { \l__enumext_tmp:n {#1} }

```

(End of definition for mark-pos, show-ans, and show-pos.)

13.27.1 Storing structure of the environments

The idea behind “*storing structure*” in the *sequence* is to have a copy of the *structure of the environment* in which the key `save-ans` is being executed so we must capture the *optional argument* passed to the levels of the environment in which it is executed and “*storing*” this in the *sequence*.

```

\__enumext_store_active_keys:n
\__enumext_store_active_keys_vii:n

```

The functions `__enumext_store_active_keys:n` and `__enumext_store_active_keys_vii:n` will be responsible for the “*storing keys*” filtered from the *optional argument* of the environment in which the key `save-ans` is executed and the levels within this for the `enumext` and `enumext*` environments. We will execute this function only if the variable `\l__enumext_store_save_key_X_bool` is false, that is, the key `store-key` is not active, establishing the variable `\l__enumext_store_save_key_X_tl` with the filtered *<keys>*.

```

2348 \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2349 {
2350   \bool_if:cF { l__enumext_store_save_key_ \__enumext_level: _bool }
2351   {
2352     \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2353     \tl_set:ce
2354       { l__enumext_store_save_key_ \__enumext_level: _tl }
2355       { \__enumext_filter_save_key:n {#1} }
2356   }
2357 }
2358 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2359 {
2360   \bool_if:NF \l__enumext_store_save_key_vii_bool
2361   {
2362     \tl_clear:N \l__enumext_store_save_key_vii_tl
2363     \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2364   }
2365 }

```

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

13.27.2 Setting save-key key

Since this “*storing structure*” in the *sequence* established by the `save-ans` key when executing `\anskey` or `anskey*`, we will not be able to modify it. The best thing here is to have a key that allows you to modify the *optional argument* of the “*storing structure*” in the *sequence*.

`save-key` The values set by this key passed in the *optional argument* of the `enumext` and `enumext*` environments will override the values of the `\l__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.

```

2366 \cs_set_protected:Npn \__enumext_tmp:n #1
2367   {
2368     \keys_define:nn { enumext / enumext* }
2369     {
2370       save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2371       save-key .value_required:n = true,
2372     }
2373     \keys_define:nn { enumext / #1 }
2374     {
2375       save-key .code:n = \__enumext_parse_save_key:n {##1},
2376       save-key .value_required:n = true,
2377     }
2378   }
2379 \clist_map_inline:nn { level-1, level-2, level-3, level-4 } { \__enumext_tmp:n {#1} }

```

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

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

```

2380 \cs_new_protected:Npn \__enumext_parse_save_key:n #1
2381   {
2382     \bool_set_true:c { l__enumext_store_save_key_ \__enumext_level: _bool }
2383     \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2384     \tl_set:ce
2385       { l__enumext_store_save_key_ \__enumext_level: _tl }
2386       { \__enumext_filter_save_key:n {#1} }
2387   }
2388 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2389   {
2390     \bool_set_true:N \l__enumext_store_save_key_vii_bool
2391     \tl_clear:N \l__enumext_store_save_key_vii_tl
2392     \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2393   }

```

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

13.27.3 Internal functions to store optional arguments

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

```

2394 \cs_new:Npn \__enumext_filter_save_key:n #1
2395   {
2396     \use:e
2397     {
2398       \keyval_parse:NNn
2399         \__enumext_filter_save_key_key:n
2400         \__enumext_filter_save_key_pair:nn {#1}
2401     }
2402   }

```

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.

```

2403 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2404   {
2405     \str_case:nnF {#1}
2406     {
2407       { resume } {} { resume* } {} { no-store } {} { base-fix } {}
2408     }
2409     { , { \exp_not:n {#1} } }
2410   }

```

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

```

2411 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2412 {
2413   \str_case:nnF {#1}
2414   {
2415     { series } {} { resume } {} { save-ans } {} { save-ref } {}
2416     { save-key } {} { check-ans } {} { show-ans } {} { show-pos } {}
2417     { wrap-ans } {} { mark-ans } {} { wrap-opt } {} { save-sep } {}
2418     { mark-ref } {} { mini-env } {} { mini-sep } {} { mini-right } {}
2419     { mini-right* } {}
2420   }
2421   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2422 }

```

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

13.27.4 Function for storing content in prop list

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

`__enumext_store_addto_prop:V`

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

```

2423 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2424 {
2425   \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2426   {
2427     \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2428   }
2429   { #1 }
2430 }
2431 \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V }

```

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

13.27.5 Function for storing content in sequence

`__enumext_store_addto_seq:n` The function `__enumext_store_addto_seq:n` stores the `{\content}` in *sequence* defined by `save-ans` key. This function is used by `\anskey` in `enumext`, `\item*` in `keyans` and `\anspic` in `keyanspic`.

`__enumext_store_addto_seq:v`

`__enumext_store_addto_seq:V`

The form in which the `{\content}` 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.

```

2432 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2433 {
2434   \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
2435 }
2436 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V }

```

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

13.27.6 Functions for storing structure in the sequence

`__enumext_store_level_open:` 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.

`__enumext_store_level_close:`

```

2437 \cs_new_protected:Nn \__enumext_store_level_open:
2438 {
2439   \bool_if:NT \l__enumext_check_answers_bool
2440   {
2441     \tl_if_empty:cTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2442     {
2443       \__enumext_store_addto_seq:n
2444       {
2445         \item \begin{enumext}
2446       }
2447     }
2448     {
2449       \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2450       {

```

```

2451         \item \begin{enumext} [
2452         ]
2453         \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2454         {
2455         ]
2456         }
2457         \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2458     }
2459 }
2460 }
2461 \cs_new_protected:Nn \__enumext_store_level_close:
2462 {
2463     \bool_if:NT \l__enumext_check_answers_bool
2464     {
2465         \__enumext_store_addto_seq:n { \end{enumext} }
2466     }
2467 }

```

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

__enumext_store_level_open_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.

```

2468 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2469 {
2470     \bool_if:NT \l__enumext_check_answers_bool
2471     {
2472         \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2473         {
2474             \__enumext_store_addto_seq:n
2475             {
2476                 \item \begin{enumext*}
2477             }
2478         }
2479         {
2480             \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2481             {
2482                 \item \begin{enumext*}[
2483             ]
2484             \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2485             {
2486                 ]
2487             }
2488             \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2489         }
2490     }
2491 }
2492 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2493 {
2494     \bool_if:NT \l__enumext_check_answers_bool
2495     {
2496         \__enumext_store_addto_seq:n { \end{enumext*} }
2497     }
2498 }

```

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

13.27.7 Function for show marks and position

__enumext_print_keyans_box:NN 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

```

2499 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2500 {
2501     \mode_leave_vertical:
2502     \skip_horizontal:n { -\dim_use:N #2 }
2503     \makebox[0pt][ r ]
2504     {
2505         \makebox[ \dim_use:N #1 ] [ \l__enumext_mark_position_str ]
2506         {
2507             \tl_use:N \l__enumext_mark_answer_sym_tl

```

```

2508     }
2509   }
2510   \skip_horizontal:n { \dim_use:N #2 }
2511 }
2512 \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`.

```

2513 \cs_new_protected:Nn \__enumext_store_internal_ref:
2514 {
2515   \cs_set_protected:Npn \__enumext_tmp:n ##1
2516   {
2517     \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2518     \tl_reverse:c { l__enumext_label_copy_##1_tl }
2519     \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2520     \tl_reverse:c { l__enumext_label_copy_##1_tl }
2521   }
2522   \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2523   \cs_set:Npn \__enumext_tmp:n ##1
2524   { . \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.

```

2525 \bool_lazy_all:nT
2526 {
2527   { \bool_if_p:N \g__enumext_starred_bool }
2528   { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2529 }
2530 {
2531   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2532   { \tl_use:N \l__enumext_label_copy_vii_tl }
2533 }
2534 \bool_lazy_all:nT
2535 {
2536   { \bool_not_p:n { \g__enumext_standar_bool } }
2537   { \bool_if_p:N \l__enumext_standar_bool }
2538   { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2539 }
2540 {
2541   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2542   {
2543     \tl_use:N \l__enumext_label_copy_vii_tl
2544     \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2545   }
2546 }

```

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.

```

2547 \bool_lazy_all:nT
2548 {
2549   { \bool_if_p:N \g__enumext_standar_bool }
2550   { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2551   { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2552 }
2553 {
2554   \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2555   {
2556     \tl_use:N \l__enumext_label_copy_i_tl
2557     \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2558   }
2559 }
2560 \cs_set:Npn \__enumext_tmp:n ##1
2561 { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } . }
2562 \bool_lazy_all:nT

```



```

2563     {
2564       { \bool_if_p:N \g__enumext_standar_bool }
2565       { \bool_if_p:N \l__enumext_starred_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         \int_step_function:nnN { 1 } { \l__enumext_level_int } \l__enumext_tmp:n
2572         \tl_use:N \l__enumext_label_copy_vii_tl
2573       }
2574     }

```

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

```

2575     \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2576     {
2577       \l__enumext_store_name_tl \c_colon_str
2578       \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2579     }

```

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.

```

2580     \tl_put_right:Ne \l__enumext_write_aux_file_tl
2581     {
2582       \__enumext_newlabel:nn
2583       { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2584       { \l__enumext_newlabel_arg_two_tl }
2585     }
2586     \l__enumext_write_aux_file_tl
2587   }

```

(End of definition for `__enumext_store_internal_ref:`)

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

`__enumext_store_anskey_code:n`

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

```

2588     \cs_new_protected:Npn \__enumext_store_anskey_code:n #1
2589     {
2590       \int_gincr:N \g__enumext_item_anskey_int
2591       \__enumext_store_addto_prop:n {#1}
2592       \bool_if:NT \l__enumext_store_ref_key_bool
2593       {
2594         \__enumext_store_internal_ref:
2595       }
2596       \__enumext_anskey_show_wrap_left:n { #1 }

```

Now we start processing the $\langle\langle key = val \rangle\rangle$ 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 $\langle\langle keys \rangle\rangle$, if the *break-col* key is present and the command is running under `enumext` (not in `enumext*`) we will add `\columnbreak` and then `\item`.

```

2597     \tl_clear:N \l__enumext_store_anskey_arg_tl
2598     \bool_lazy_and:nnT
2599     { \bool_if_p:N \l__enumext_store_columns_break_bool }
2600     { \bool_not_p:n { \l__enumext_starred_bool } }
2601     {
2602       \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2603     }
2604     \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 $\langle\langle number \rangle\rangle$ to `\l__enumext_store_anskey_arg_tl`.

```

2605     \bool_lazy_and:nnT
2606     { \bool_not_p:n { \l__enumext_starred_bool } }
2607     { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2608     {
2609       \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2610       {
2611         ( \exp_not:V \l__enumext_store_item_join_int )
2612       }
2613     }

```

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

```

2614 \bool_if:NTF \l__enumext_store_item_star_bool
2615 {
2616   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2617   \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2618   {
2619     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2620     {
2621       [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2622     }
2623   }
2624   \dim_compare:nT
2625   {
2626     \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2627   }
2628   {
2629     \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2630     {
2631       [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2632     }
2633   }
2634   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2635 }
2636 {
2637   \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2638 }

```

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`.

```

2639 \bool_lazy_and:nnT
2640 { \bool_if_p:N \l__enumext_store_ref_key_bool }
2641 { \bool_if_p:N \l__enumext_hyperref_bool }
2642 {
2643   \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2644   {
2645     \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2646     { \exp_not:V \l__enumext_mark_ref_sym_tl }
2647   }
2648 }
2649 \l__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2650 }

```

(End of definition for `\l__enumext_store_anskey_code:n`.)

`\l__enumext_anskey_show_wrap_arg:n`

The function `\l__enumext_anskey_show_wrap_arg:n` “wraps” the `{⟨argument⟩}` passed to `\anskey` and the `(body)` for `anskey*` when using the `wrap-ans` key.

```

2651 \cs_new_protected:Npn \l__enumext_anskey_show_wrap_arg:n #1
2652 {
2653   \par
2654   \bool_if:NTF \l__enumext_starred_bool
2655   {
2656     \l__enumext_print_keyans_box:NN
2657     \l__enumext_labelwidth_vii_dim \l__enumext_labelsep_vii_dim
2658   }
2659   {
2660     \l__enumext_print_keyans_box:cc
2661     { \l__enumext_labelwidth_ \l__enumext_level: _dim }
2662     { \l__enumext_labelsep_ \l__enumext_level: _dim }
2663   }
2664   \l__enumext_anskey_wrapper:n { #1 }
2665 }

```

(End of definition for `\l__enumext_anskey_show_wrap_arg:n`.)

`\l__enumext_anskey_show_wrap_left:n`

The function `\l__enumext_anskey_show_wrap_left:n` will show the “`mark`” defined by the `mark-ans` key or the “`position`” of the `{⟨content⟩}` stored in the `prop list` when using the `show-pos` key on the left margin next to the “wraps” `{⟨argument⟩}` passed to `\anskey` and the `(body)` in `anskey*` on the right side when using the `show-ans` key.

```

2666 \cs_new_protected:Npn \l__enumext_anskey_show_wrap_left:n #1
2667 {

```

```

2668 \bool_if:NT \l__enumext_show_answer_bool
2669 {
2670   \__enumext_anskey_show_wrap_arg:n { #1 }
2671 }
2672 \bool_if:NT \l__enumext_show_position_bool
2673 {
2674   \tl_set:Nc \l__enumext_mark_answer_sym_tl
2675   {
2676     \group_begin:
2677     \exp_not:N \normalfont
2678     \exp_not:N \footnotesize [ \int_eval:n
2679     {
2680       \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2681     }
2682     ]
2683     \group_end:
2684   }
2685   \__enumext_anskey_show_wrap_arg:n { #1 }
2686 }
2687 }

```

(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*`.

```

\__enumext_anskey_unknown:n
\__enumext_anskey_unknown:n
2688 \keys_define:nn { enumext / anskey }
2689 {
2690   break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2691   break-col .default:n = true,
2692   break-col .value_forbidden:n = true,
2693   item-join .int_set:N = \l__enumext_store_item_join_int,
2694   item-join .value_required:n = true,
2695   item-star .bool_set:N = \l__enumext_store_item_star_bool,
2696   item-star .default:n = true,
2697   item-star .value_forbidden:n = true,
2698   item-sym* .tl_set:N = \l__enumext_store_item_symbol_tl,
2699   item-sym* .value_required:n = true,
2700   item-pos* .dim_set:N = \l__enumext_store_item_symbol_sep_dim,
2701   item-pos* .value_required:n = true,
2702   unknown .code:n = { \__enumext_anskey_unknown:n {#1} },
2703 }

```

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`.

```

2704 \cs_new_protected:Npn \__enumext_anskey_unknown:n #1
2705 {
2706   \exp_args:NV \__enumext_anskey_unknown:nn \l_keys_key_str {#1}
2707 }
2708 \cs_new_protected:Npn \__enumext_anskey_unknown:nn #1 #2
2709 {
2710   \tl_if_blank:nTF {#2}
2711   {
2712     \msg_error:nnn { enumext } { anskey-cmd-key-unknown } {#1}
2713   }
2714   {
2715     \msg_error:nnnn { enumext } { anskey-cmd-key-value-unknown } {#1} {#2}
2716   }
2717 }

```

(End of definition for __enumext_anskey_unknown:n and __enumext_anskey_unknown:nn.)

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

`\anskey` We will first call the function `__enumext_anskey_safe_outer:` to be sure where we execute the command, then we will check the state of the variable `\l__enumext_check_answers_bool` set by the key `no-store`, if is true we will increment `\g__enumext_item_anskey_int` for the internal “*check answer*” system and

execute the function `__enumext_anskey_safe_inner:n` to ensure that the command is not nested and that the argument is not empty, finally search the `[{key = val}]` and call the function `__enumext_store_anskey_code:n`.

```

2718 \NewDocumentCommand \anskey { o +m }
2719 {
2720   \__enumext_anskey_safe_outer:
2721   \group_begin:
2722     \bool_if:NT \l__enumext_check_answers_bool
2723     {
2724       \tl_if_novalue:nF {#1}
2725       {
2726         \keys_set:nn { enumext / anskey } {#1}
2727       }
2728       \tl_if_blank:nTF {#2}
2729       {
2730         \msg_error:nn { enumext } { anskey-empty-arg }
2731       }
2732       {
2733         \__enumext_anskey_safe_inner:
2734         \__enumext_store_anskey_code:n {#2}
2735       }
2736     }
2737   \group_end:
2738 }

```

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

13.30.1 Internal functions for the command

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

`__enumext_anskey_safe_inner:`

```

2739 \cs_new_protected:Nn \__enumext_anskey_safe_outer:
2740 {
2741   \bool_if:NF \l__enumext_store_active_bool
2742   {
2743     \msg_error:nnnn { enumext } { anskey-wrong-place } { anskey } { enumext }
2744   }
2745   \int_compare:nNt { \l__enumext_keyans_level_int } = { 1 }
2746   {
2747     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans }
2748   }
2749   \int_compare:nNt { \l__enumext_keyans_level_h_int } = { 1 }
2750   {
2751     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyans* }
2752   }
2753   \int_compare:nNt { \l__enumext_keyans_pic_level_int } = { 1 }
2754   {
2755     \msg_error:nnnn { enumext } { command-wrong-place } { anskey } { keyanspic }
2756   }
2757 }

```

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.

```

2758 \cs_new_protected:Nn \__enumext_anskey_safe_inner:
2759 {
2760   \int_incr:N \l__enumext_anskey_level_int
2761   \int_compare:nNt { \l__enumext_anskey_level_int } > { 1 }
2762   {
2763     \msg_error:nn { enumext } { anskey-nested }
2764   }
2765   \bool_if:NF \l__enumext_item_number_bool
2766   {
2767     \msg_error:nn { enumext } { anskey-unnumber-item }
2768   }
2769   \mode_if_math:T
2770   {
2771     \msg_error:nne { enumext } { anskey-math-mode } { \c_backslash_str anskey }
2772   }
2773 }

```

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

13.31 The environment anskey*

Managing *verbatim content* in an environment is quite complicated, I learned that when creating the `scontents` package, so to be able to have support at this point it is best to play a little with the internal code of `scontents` and `hooks`. Some considerations I should have here before implementing this:

- If some package, class or user has defined the environment with the same name somewhere in the document it would be a problem, you would not know what argument has been passed to `store-env`, if you are using the key `print-env` or the `write-out` key, sure, I can detect and modify it within the `enumext` and `enumext*` environments, but it would look strange not to have some keys available when running within these environments.
- A better (perhaps a bit paranoid) option is to define it within the environment in which the `save-ans` key is executed. and have it available only when that key is executed, here I would have absolute control of the `(keys)` and I make sure that `write-out` is not used, then using `hooks after` I undefine it and using `hook before` I check if it has been created by any package, class or user and I return a error, then the user will have to see how to solve the problem.

`__enumext_undefine_anskey_env:` The function `__enumext_undefine_anskey_env:` will undefine the environment `anskey*` and will be passed to the function `__enumext_execute_after_env:` (§13.32) which is executed after the environment in which the key `save-ans` is active.

```

2774 \cs_new_protected:Nn \__enumext_undefine_anskey_env:
2775 {
2776   \cs_undefine:c { anskey* }
2777   \cs_undefine:c { endanskey* }
2778   \cs_undefine:c { __scontents_anskey*_env_begin: }
2779   \cs_undefine:c { __scontents_anskey*_env_end: }
2780 }

```

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

```

2781 \__enumext_before_env:nn { enumext }
2782 {
2783   \bool_lazy_and:nnT
2784     { \int_compare_p:nNn { \__enumext_level_int } = { 0 } }
2785     { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
2786     {
2787       \cs_if_free:cF { __scontents_anskey*_env_begin: }
2788       {
2789         \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2790       }
2791     }
2792 }
2793 \__enumext_before_env:nn { enumext* }
2794 {
2795   \bool_lazy_and:nnT
2796     { \int_compare_p:nNn { \__enumext_level_int } = { 0 } }
2797     { \int_compare_p:nNn { \__enumext_level_h_int } = { 0 } }
2798     {
2799       \cs_if_free:cF { __scontents_anskey*_env_begin: }
2800       {
2801         \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2802       }
2803     }
2804 }

```

Detection of the `anskey*` environment inside the `keyans`, `keyans*` and `keyanspic` environments, if preceded by a not numbered `\item` or if it is in *math mode* returning the appropriate messages.

```

2805 \__enumext_before_env:nn { anskey* }
2806 {
2807   \int_compare:nNnT { \__enumext_keyans_level_int } = { 1 }
2808   {
2809     \msg_error:nnn { enumext } { anskey-env-wrong } { keyans }
2810   }
2811   \int_compare:nNnT { \__enumext_keyans_level_h_int } = { 1 }
2812   {
2813     \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
2814   }
2815   \int_compare:nNnT { \__enumext_keyans_pic_level_int } = { 1 }
2816   {
2817     \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
2818   }
2819   \bool_if:NF \__enumext_item_number_bool
2820   {

```

```

2821     \msg_error:nn { enumext } { anskey-unnumber-item }
2822   }
2823   \mode_if_math:T
2824   {
2825     \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
2826   }
2827 }

```

(End of definition for `__enumext_undefine_anskey_env:`)

anskey* The function `__enumext_anskey_env_make:n` creates the environment **anskey*** (custom version of `scontents` environment) by setting the initial keys `store-env={⟨store name⟩}` and `print-env=false`.

`__enumext_anskey_env_make:n`
`__enumext_anskey_env_define_keys:`
`__enumext_anskey_env_undefine_keys:`
`__enumext_anskey_env_undefine_keys:`
`__enumext_anskey_env_reset_keys:`
`__enumext_rescan_anskey_env:n`

To maintain the *scope* of the environment and that it is only active when the key `save-ans` is active we will pass this function to the function `__enumext_storing_exec:` (§13.26.1) and we will execute it only if the variable `__enumext_anskey_env_bool` is true, with this we prevent it from being executed again when the environment is nested and the key `save-ans` is active, which returns an error for part of the package **scontents**.

```

2828 \cs_new_protected:Npn \__enumext_anskey_env_make:n #1
2829 {
2830   \bool_if:NT \__enumext_anskey_env_bool
2831   {
2832     \newenvsc{anskey*}[store-env=#1,print-env=false]
2833     \__enumext_anskey_env_exec:
2834   }
2835 }
2836 \cs_generate_variant:Nn \__enumext_anskey_env_make:n { V }

```

The function `__enumext_anskey_env_define_keys:` will add the keys `break-col`, `item-join`, `item-join`, `item-star`, `item-sym*` and `item-pos*` and will leave the keys `print-env`, `store-env` and `write-out` undefined. We will apply this function using the *hook* function `__enumext_before_env:nn`.

```

2837 \cs_new_protected:Nn \__enumext_anskey_env_define_keys:
2838 {
2839   \keys_define:nn { scontents / scontents }
2840   {
2841     break-col .bool_gset:N = \__enumext_store_columns_break_bool,
2842     break-col .default:n = true,
2843     break-col .value_forbidden:n = true,
2844     item-join .int_gset:N = \__enumext_store_item_join_int,
2845     item-join .value_required:n = true,
2846     item-star .bool_gset:N = \__enumext_store_item_star_bool,
2847     item-star .default:n = true,
2848     item-star .value_forbidden:n = true,
2849     item-sym* .tl_gset:N = \__enumext_store_item_symbol_tl,
2850     item-sym* .value_required:n = true,
2851     item-pos* .dim_gset:N = \__enumext_store_item_symbol_sep_dim,
2852     item-pos* .value_required:n = true,
2853     print-env .undefine:,
2854     store-env .undefine:,
2855     write-out .undefine:,
2856     unknown .code:n = { \__enumext_anskey_env_undefine:n {##1} },
2857   }
2858 }

```

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_undefine:n`.

```

2859 \cs_new_protected:Npn \__enumext_anskey_env_undefine:n #1
2860 {
2861   \exp_args:NV \__enumext_anskey_env_undefine:nn \l_keys_key_str {#1}
2862 }
2863 \cs_new_protected:Npn \__enumext_anskey_env_undefine:nn #1#2
2864 {
2865   \tl_if_blank:nTF {#2}
2866   {
2867     \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
2868   }
2869   {
2870     \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
2871   }
2872 }

```

The function `__enumext_anskey_env_reset_keys:` will leave the keys `break-col`, `item-join`, `item-join`, `item-star`, `item-sym*` and `item-pos*` undefined. We will apply this function using the `hook` function `__enumext_after_env:nn`.

```

2873 \cs_new_protected:Nn \__enumext_anskey_env_reset_keys:
2874 {
2875   \keys_define:nn { scontents / scontents }
2876   {
2877     break-col .undefine:,
2878     item-join .undefine:,
2879     item-star .undefine:,
2880     item-sym* .undefine:,
2881     item-pos* .undefine:,
2882     write-out .code:n = {
2883       \bool_set_false:N \l__scontents_storing_bool
2884       \bool_set_true:N \l__scontents_writing_bool
2885       \tl_set:Nn \l__scontents_fname_out_tl {##1}
2886     },
2887     write-out .value_required:n = true,
2888     print-env .meta:nn = { scontents } { print-env = ##1 },
2889     print-env .default:n = true,
2890     store-env .meta:nn = { scontents } { store-env = ##1 },
2891     unknown .code:n = { \__scontents_parse_environment_keys:n {##1} },
2892   }
2893 }

```

The function `__enumext_rescan_anskey_env:n` will be responsible for bringing the (*body*) of the environment saved in the sequence `\g__scontents_name_(store name)_seq` to pass it to our *sequence* and *prop list*.

```

2894 \cs_new_protected:Npn \__enumext_rescan_anskey_env:n #1
2895 {
2896   \group_begin:
2897   \int_set:Nn \tex_newlinechar:D { \^^J }
2898   \__scontents_rescan_tokens:x
2899   {
2900     \endgroup % This assumes \catcode`\=0... Things might go off otherwise.
2901     #1
2902   }
2903 }

```

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

`__enumext_anskey_env_exec:` The function `__enumext_anskey_env_exec:` will be responsible for processing all the code necessary for the execution of the environment. The first thing will be to add our (*keys*).

```

2904 \cs_new_protected:Nn \__enumext_anskey_env_exec:
2905 {
2906   \__enumext_before_env:nn { anskey* }
2907   {
2908     \__enumext_anskey_env_define_keys:
2909   }

```

Now we will execute our actions after the `anskey*` environment is closed. We'll fetch the contents of the *environment body* that is now saved in `\g__scontents_name_(store name)_seq` and store it in the variable `\l__enumext_store_anskey_env_tl` then we execute the rest of the functions.

```

2910   \hook_if_empty:nF {env/anskey*/after}
2911   {
2912     \hook_gremove_code:nn {env/anskey*/after} { * }
2913   }
2914   \__enumext_after_env:nn { anskey* }
2915   {
2916     \__enumext_anskey_env_save_keys:
2917     \tl_clear:N \l__enumext_store_anskey_env_tl
2918     \tl_clear:N \l__enumext_store_anskey_opt_tl
2919     \bool_if:NT \l__enumext_check_answers_bool
2920     {
2921       \tl_gset:Ne \l__enumext_store_anskey_env_tl
2922       {
2923         \seq_item:ce { g__scontents_name_ \l__enumext_store_name_tl _seq } { -1 }
2924       }
2925       \regex_match:nVTF
2926       { ^\s* \z | ^\s* \u{c__scontents_hidden_space_str} \z }
2927       \l__enumext_store_anskey_env_tl

```



```

2928         {
2929             \msg_error:nn { enumext } { anskey-empty-arg }
2930         }
2931         {
2932             \__enumext_anskey_env_store:
2933         }
2934     }
2935     \__enumext_anskey_env_clean_vars:
2936     \__enumext_anskey_env_reset_keys:
2937 }
2938 }

```

- The use of `\hook_gremove_code:nn` is necessary here, otherwise the `{code}` passed to `__enumext_after_env:nn{anskey*}` will be accumulated for each execution. The last function `__enumext_anskey_env_reset_keys:` is necessary so as not to hinder any `scontents` environment running within `enumext` or `enumext*`.

(End of definition for `__enumext_anskey_env_exec:.`)

```

\__enumext_anskey_env_save_keys:
\__enumext_anskey_env_store:
\__enumext_anskey_env_clean_vars:

```

The function `__enumext_anskey_env_save_keys:` processing the `[key = val]` passed to the environment and save this in the variable `\l__enumext_store_anskey_opt_tl`. If the `break-col` key is present and the environment is running under `enumext` (not in `enumext*`) we will add the key `break-col`.

```

2939 \cs_new_protected:Nn \__enumext_anskey_env_save_keys:
2940 {
2941     \bool_lazy_and:nnT
2942     { \bool_if_p:N \g__enumext_store_columns_break_bool }
2943     { \bool_not_p:n { \l__enumext_starred_bool } }
2944     {
2945         \tl_put_left:Ne \l__enumext_store_anskey_opt_tl { ,break-col, }
2946     }

```

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

```

2947     \bool_lazy_and:nnT
2948     { \bool_not_p:n { \l__enumext_starred_bool } }
2949     { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }
2950     {
2951         \tl_put_left::Ne \l__enumext_store_anskey_opt_tl
2952         {
2953             ,item-join = \exp_not:V \g__enumext_store_item_join_int,
2954         }
2955     }

```

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

```

2956     \bool_if:NT \g__enumext_store_item_star_bool
2957     {
2958         \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2959         {
2960             ,item-star,
2961         }
2962         \tl_if_empty:NF \g__enumext_store_item_symbol_tl
2963         {
2964             \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2965             {
2966                 ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,
2967             }
2968         }
2969         \dim_compare:nT
2970         {
2971             \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
2972         }
2973         {
2974             \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2975             {
2976                 ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
2977             }
2978         }
2979     }
2980 }

```

The function `__enumext_anskey_env_store:` will be responsible for storing the content of the environment using the functions `__enumext_store_anskey_code:n` and `__enumext_rescan_anskey_env:n`.

```

2981 \cs_new_protected:Nn \__enumext_anskey_env_store:

```

```

2982 {
2983   \group_begin:
2984     \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
2985     {
2986       \exp_args:Ne
2987         \__enumext_store_anskey_code:n
2988         {
2989           \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2990         }
2991     }
2992     {
2993       \keys_set_known:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
2994       \exp_args:Ne
2995         \__enumext_store_anskey_code:n
2996         {
2997           \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2998         }
2999     }
3000   \group_end:
3001 }

```

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

```

3002 \cs_new_protected:Nn \__enumext_anskey_env_clean_vars:
3003 {
3004   \bool_gset_false:N \g__enumext_store_columns_break_bool
3005   \int_gzero:N \g__enumext_store_item_join_int
3006   \bool_gset_false:N \g__enumext_store_item_star_bool
3007   \tl_gclear:N \g__enumext_store_item_symbol_tl
3008   \dim_gzero:N \g__enumext_store_item_symbol_sep_dim
3009 }

```

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

13.32 Executing `anskey*`, `check-ans` and `write .log`

`__enumext_execute_after_env:`

The `__enumext_execute_after_env:` function will first return the appropriate message for the end of the environment in which the `save-ans` key is being executed, then call the `__enumext_item_answer_diff:` function and then will write the values of the global variables used to the `.log` file. If the key `check-ans` is active it will execute the function `__enumext_check_ans_show:` and show the result in the terminal, otherwise it will execute the function `__enumext_check_ans_log:` and write the results in the `.log` file, undefine the environment `anskey*` (§13.31) through the function `__enumext_undefine_anskey_env:` and finally we execute the function `__enumext_reset_global_vars:` returning the used variables to their original state.

```

3010 \cs_new_protected:Nn \__enumext_execute_after_env:
3011 {
3012   \int_compare:nNnT { \l__enumext_level_int } = { 0 }
3013   {
3014     \tl_if_empty:NF \g__enumext_store_name_tl
3015     {
3016       \__enumext_stop_save_ans_msg:
3017       \__enumext_item_answer_diff:
3018       \__enumext_log_global_vars:
3019       \__enumext_log_answer_vars:
3020       \bool_if:NTF \g__enumext_check_ans_key_bool
3021       {
3022         \__enumext_check_ans_show:
3023       }
3024       { \__enumext_check_ans_log: }
3025       \__enumext_undefine_anskey_env:
3026     }
3027     \__enumext_reset_global_vars:
3028   }
3029 }

```

(End of definition for `__enumext_execute_after_env:`.)

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

13.33 Common functions for keyans, keyans* and keyanspic

13.33.1 Storing content in prop list

`__enumext_keyans_addto_prop:n`

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

```

3030 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
3031 {
3032   \tl_clear:N \l__enumext_store_current_label_tl
3033   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
3034   {
3035     \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
3036   }
3037   {
3038     \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
3039   }

```

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

```

3040   \tl_if_novalue:nF { #1 }
3041   {
3042     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
3043     {
3044       \tl_put_right:Ne \l__enumext_store_current_label_tl
3045       {
3046         \l__enumext_store_keyans_item_opt_sep_tl
3047       }
3048     }
3049     \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
3050   }
3051   \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
3052 }

```

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

`__enumext_keyans_store_ref:`

`__enumext_keyans_store_ref_aux_i:`

`__enumext_keyans_store_ref_aux_ii:`

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

```

3053 \cs_new_protected:Nn \__enumext_keyans_store_ref:
3054 {
3055   \bool_if:NT \l__enumext_store_ref_key_bool
3056   {
3057     \cs_set_protected:Npn \__enumext_tmp:n ##1
3058     {
3059       \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
3060       \tl_reverse:c { l__enumext_label_copy_##1_tl }
3061       \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
3062       \tl_reverse:c { l__enumext_label_copy_##1_tl }
3063     }
3064     \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
3065     \__enumext_keyans_store_ref_aux_i:
3066   }
3067 }

```

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

```

3068 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
3069 {
3070   \bool_if:NT \g__enumext_starred_bool
3071   {
3072     \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
3073   }
3074   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }

```

```

3075     {
3076       \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
3077       { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
3078     }
3079     \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
3080     {
3081       \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
3082       { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
3083     }
3084     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
3085     {
3086       \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
3087       { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
3088     }
3089     \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
3090     {
3091       \l__enumext_store_name_tl \c_colon_str
3092       \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
3093     }
3094     \__enumext_keyans_store_ref_aux_ii:
3095   }

```

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.

```

3096 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
3097 {
3098   \tl_put_right:Ne \l__enumext_write_aux_file_tl
3099   {
3100     \__enumext_newlabel:nn
3101     { \exp_not:V \l__enumext_newlabel_arg_one_tl }
3102     { \l__enumext_newlabel_arg_two_tl }
3103   }
3104   \l__enumext_write_aux_file_tl
3105 }

```

(End of definition for `__enumext_keyans_store_ref:`, `__enumext_keyans_store_ref_aux_ii:`, 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.

```

3106 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
3107 {
3108   \tl_clear:N \l__enumext_store_current_label_tl
3109   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
3110   {
3111     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
3112   }
3113   {
3114     \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
3115   }
3116   \tl_if_novalue:nF { #1 }
3117   {
3118     \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
3119     {
3120       \tl_put_right:Ne \l__enumext_store_current_label_tl
3121       {
3122         \l__enumext_store_keyans_item_opt_sep_tl
3123       }
3124     }
3125     \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
3126   }
3127   \__enumext_keyans_addto_seq_link:
3128 }

```

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.

```

3129 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
3130 {
3131   \bool_lazy_and:nnT
3132     { \bool_if_p:N \l__enumext_store_ref_key_bool }
3133     { \bool_if_p:N \l__enumext_hyperref_bool }
3134     {
3135       \tl_put_right:Ne \l__enumext_store_current_label_tl
3136       {
3137         \hfill \exp_not:N \hyperlink
3138         {
3139           \exp_not:V \l__enumext_newlabel_arg_one_tl
3140         }
3141         { \exp_not:V \l__enumext_mark_ref_sym_tl }
3142       }
3143     }
3144     \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
3145     \bool_if:NT \l__enumext_check_answers_bool
3146     {
3147       \int_gincr:N \g__enumext_item_anskey_int
3148     }
3149 }

```

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

<pre> __enumext_keyans_show_left:n __enumext_keyans_show_ans: __enumext_keyans_show_pos: __enumext_keyans_show_item_opt: </pre>	<p>Common function to show <i>starred commands</i> <code>\item*</code> and <code><position></code> of stored content in <i>prop list</i> for <code>keyans</code> and <code>keyanspic</code>. Need add <code>1</code> to <code>\g__enumext_{(store name)}_prop</code> for <code>show-pos</code> key.</p> <pre> 3150 \cs_new_protected:Npn __enumext_keyans_show_left:n #1 3151 { 3152 \tl_if_novalue:nF { #1 } 3153 { 3154 \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 } 3155 } 3156 \bool_if:NT \l__enumext_show_answer_bool 3157 { 3158 __enumext_keyans_show_ans: 3159 } 3160 \bool_if:NT \l__enumext_show_position_bool 3161 { 3162 __enumext_keyans_show_pos: 3163 } 3164 } 3165 \cs_new_protected:Nn __enumext_keyans_show_item_opt: 3166 { 3167 \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl 3168 { 3169 \bool_lazy_or:nnT 3170 { \bool_if_p:N \l__enumext_show_answer_bool } 3171 { \bool_if_p:N \l__enumext_show_position_bool } 3172 { 3173 __enumext_keyans_wrapper_opt:n { \l__enumext_store_current_opt_arg_tl } \c_space_tl 3174 } 3175 } 3176 } 3177 \cs_new_protected:Nn __enumext_keyans_show_ans: 3178 { 3179 \bool_if:NT \l__enumext_starred_bool 3180 { 3181 \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim 3182 \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim 3183 } 3184 \tl_put_left:Nn \l__enumext_label_v_tl 3185 { 3186 __enumext_print_keyans_box:NN 3187 \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim 3188 } </pre>
---	--

```

3189 }
3190 \cs_new_protected:Nn \__enumext_keyans_show_pos:
3191 {
3192   \bool_if:NT \l__enumext_starred_bool
3193   {
3194     \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
3195     \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
3196   }
3197   \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
3198   {
3199     \tl_set:Ne \l__enumext_mark_answer_sym_tl
3200     {
3201       \group_begin:
3202       \exp_not:N \normalfont
3203       \exp_not:N \footnotesize [ \int_eval:n
3204         {
3205           \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3206         }
3207       ]
3208       \group_end:
3209     }
3210   }
3211   {
3212     \tl_set:Ne \l__enumext_mark_answer_sym_tl
3213     {
3214       \group_begin:
3215       \exp_not:N \normalfont
3216       \exp_not:N \footnotesize [ \int_eval:n
3217         {
3218           \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
3219         }
3220       ]
3221       \group_end:
3222     }
3223   }
3224   \tl_put_left:Nn \l__enumext_label_v_tl
3225   {
3226     \__enumext_print_keyans_box:NN
3227     \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3228   }
3229 }

```

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

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

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

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

The `\item` and `\item[⟨custom⟩]` commands work in the usual way on `enumext` and we will add `\item*`, `\item*[⟨symbol⟩]` and `\item*[⟨symbol⟩][⟨offset⟩]`.

`__enumext_default_item:n` First we will see if the *optional argument* is present, if it is NOT present we will check the state of the variable `\l__enumext_check_answers_bool` set by the key `no-store`, set the boolean variable `\l__enumext_wrap_label_X_bool` to “true” for the key `wrap-label` and execute `__enumext_item_std:w` and the key `itemindent`, otherwise we will check the state of the boolean variable `\l__enumext_wrap_label_opt_X_bool` set by the key `wrap-label*` and execute `__enumext_item_std:w` with the *optional argument* and the key `itemindent`.

```

3230 \cs_new_protected:Npn \__enumext_default_item:n #1
3231 {
3232   \tl_if_novalue:nTF {#1}
3233   {
3234     \bool_if:NT \l__enumext_check_answers_bool
3235     {
3236       \int_gincr:N \g__enumext_item_number_int
3237       \bool_set_true:N \l__enumext_item_number_bool

```

```

3238     }
3239     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
3240     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3241   }
3242   {
3243     \bool_set_eq:cc
3244     { l__enumext_wrap_label_ \__enumext_level: _bool }
3245     { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
3246     \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3247   }
3248 }

```

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

`__enumext_starred_item:nn` The `\item*`, `\item*[\langle symbol \rangle]` and `\item*[\langle symbol \rangle][\langle offset \rangle]` 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* `\langle offset \rangle`.

#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`.

```

3249 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
3250 {
3251   \tl_if_novalue:nTF {#1}
3252   {
3253     \tl_gset_eq:Nc
3254     \g__enumext_item_symbol_aux_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
3255   }
3256   {
3257     \tl_gset:Nn \g__enumext_item_symbol_aux_tl {#1}
3258   }
3259   \tl_if_novalue:nTF {#2}
3260   {
3261     \dim_set_eq:cc
3262     { l__enumext_item_symbol_sep_ \__enumext_level: _dim }
3263     { l__enumext_labelsep_ \__enumext_level: _dim }
3264   }
3265   {
3266     \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
3267   }
3268   \bool_if:NT \l__enumext_check_answers_bool
3269   {
3270     \int_gincr:N \g__enumext_item_number_int
3271     \bool_set_true:N \l__enumext_item_number_bool
3272   }
3273   \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
3274   \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3275 }

```

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

```

3276 \cs_new_protected:Nn \__enumext_item_star_exec:
3277 {
3278   \tl_if_empty:cF { l__enumext_item_symbol_ \__enumext_level: _tl }
3279   {
3280     \mode_leave_vertical:
3281     \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3282     \hbox_overlap_left:n { \g__enumext_item_symbol_aux_tl }
3283     \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3284   }
3285 }

```

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

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


```

3286 \cs_new_protected:Nn \__enumext_redefine_item:
3287 {
3288   \RenewDocumentCommand \item { s o o }
3289   {
3290     \bool_if:nTF {##1}
3291     {
3292       \__enumext_starred_item:nn {##2} {##3}
3293     }
3294     { \__enumext_default_item:n {##2} }
3295   }
3296 }

```

(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).

```

3297 \cs_new_protected:Nn \__enumext_make_label:
3298 {
3299   \IfDocumentMetadataTF
3300   {
3301     \__enumext_make_label_box:
3302   }
3303   {
3304     \bool_if:NTF \l__enumext_mode_box_bool
3305     {
3306       \__enumext_make_label_box:
3307     }
3308     {
3309       \__enumext_make_label_std:
3310     }
3311   }
3312 }

```

Standard definition when `\DocumentMetadata` is not active.

```

3313 \cs_new_protected:Nn \__enumext_make_label_std:
3314 {
3315   \RenewDocumentCommand \makelabel { m }
3316   {
3317     \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
3318     \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3319     \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3320     {
3321       \__enumext_item_star_exec:
3322       \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3323     }
3324     { ##1 }
3325     \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
3326     \tl_gclear:N \g__enumext_item_symbol_aux_tl
3327   }
3328 }

```

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`.

```

3329 \cs_new_protected:Nn \__enumext_make_label_box:
3330 {
3331   \RenewDocumentCommand \makelabel { m }
3332   {
3333     \strut\smash
3334     {
3335       \makebox
3336       [ \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim } ]
3337       [ \str_use:c { l__enumext_align_label_pos_ \__enumext_level: _str } ]
3338       {
3339         \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3340         \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3341         {
3342           \__enumext_item_star_exec:
3343           \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }

```

```

3344         }
3345         { ##1 }
3346         \tl_gclear:N \g__enumext_item_symbol_aux_tl
3347     }
3348     } % close smash
3349 }
3350 }

```

(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*
3351 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
3352 {
3353     \keys_define:nn { enumext / #1 }
3354     {
3355         item-sym* .tl_set:c = { \__enumext_item_symbol_#2_tl },
3356         item-sym* .value_required:n = true,
3357         item-sym* .initial:n = {\textasteriskcentered},
3358         item-pos* .dim_set:c = { \__enumext_item_symbol_sep_#2_dim },
3359         item-pos* .value_required:n = true,
3360     }
3361 }
3362 \clist_map_inline:nn
3363 {
3364     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
3365 }
3366 { \__enumext_tmp:nn #1 }

```

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

13.36 Handling unknown keys

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

13.36.1 Handling unknown keys for `keyans`, `keyans*` and `keyanspic`

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

```

\__enumext_keyans_unknown_keys:n
\__enumext_keyans_unknown_keys:nn
3367 \cs_set_protected:Npn \__enumext_tmp:n #1
3368 {
3369     \keys_define:nn { enumext / #1 }
3370     {
3371         unknown .code:n = { \__enumext_keyans_unknown_keys:n {##1} }
3372     }
3373 }
3374 \clist_map_inline:nn { keyans, keyans*, keyanspic } { \__enumext_tmp:n {#1} }

```

Internal functions for handling `unknown` key.

```

3375 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:n #1
3376 {
3377     \exp_args:NV \__enumext_keyans_unknown_keys:nn \l_keys_key_str {#1}
3378 }
3379 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:nn #1#2
3380 {
3381     \tl_if_blank:nTF {#2}
3382     {
3383         \msg_error:nnn { enumext } { keyans-unknown-key } {#1}
3384     }
3385     {
3386         \msg_error:nnnn { enumext } { keyans-unknown-key-value } {#1} {#2}
3387     }
3388 }

```

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

13.36.2 Handling unknown keys for enumext*

unknown Define and set unknown key for enumext* environment.

```

3389 \__enumext_starred_unknown_keys:n
\__enumext_starred_unknown_keys:nn
3390 {
3391   unknown .code:n = { \__enumext_starred_unknown_keys:n {#1} }
3392 }

```

Internal functions for handling unknown key.

```

3393 \cs_new_protected:Npn \__enumext_starred_unknown_keys:n #1
3394 {
3395   \exp_args:NV \__enumext_starred_unknown_keys:nn \l_keys_key_str {#1}
3396 }
3397 \cs_new_protected:Npn \__enumext_starred_unknown_keys:nn #1#2
3398 {
3399   \tl_if_blank:nTF {#2}
3400   {
3401     \msg_error:nnn { enumext } { starred-unknown-key } {#1}
3402   }
3403   {
3404     \msg_error:nnnn { enumext } { starred-unknown-key-value } {#1} {#2}
3405   }
3406 }

```

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

13.36.3 Handling unknown keys for enumext

unknown Defines and set the key unknown for enumext environment.

```

\__enumext_standar_unknown_keys:n
\__enumext_standar_unknown_keys:nn
3407 \cs_set_protected:Npn \__enumext_tmp:n #1
3408 {
3409   \keys_define:nn { enumext / #1 }
3410   {
3411     unknown .code:n = { \__enumext_standar_unknown_keys:n {##1} }
3412   }
3413 }
3414 \clist_map_inline:nn { level-1,level-2,level-3,level-4 } { \__enumext_tmp:n {#1} }

```

Internal functions for handling unknown key.

```

3415 \cs_new_protected:Npn \__enumext_standar_unknown_keys:n #1
3416 {
3417   \exp_args:NV \__enumext_standar_unknown_keys:nn \l_keys_key_str {#1}
3418 }
3419 \cs_new_protected:Npn \__enumext_standar_unknown_keys:nn #1#2
3420 {
3421   \tl_if_blank:nTF {#2}
3422   {
3423     \msg_error:nnn { enumext } { standar-unknown-key } {#1}
3424   }
3425   {
3426     \msg_error:nnnn { enumext } { standar-unknown-key-value } {#1} {#2}
3427   }
3428 }

```

(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`.

```

3429 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
3430 {
3431   \tl_if_novalue:nTF { #1 }
3432   {
3433     \bool_set_true:N \l__enumext_wrap_label_v_bool
3434     \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
3435   }
3436   {
3437     \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
3438     \__enumext_item_std:w {#1} \tl_use:N \l__enumext_fake_item_indent_v_tl

```

```

3439     }
3440 }

```

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

`__enumext_keyans_starred_item:n`

The function `__enumext_keyans_starred_item:n` which will make a temporary copy of the current `<label>`, execute the `show-ans` or `show-pos` keys using the function `__enumext_keyans_show_left:n` and will display the `<contents>` of that item using the internal copy `__enumext_item_std:w`, this is necessary to prevent incrementing the current “counter” of the original `<label>`, followed by this it will execute function `__enumext_keyans_show_item_opt:` handled by `wrap-opt` key.

```

3441 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
3442 {
3443   \tl_set_eq:NN \l__enumext_store_current_label_tmp_tl \l__enumext_label_v_tl
3444   \__enumext_keyans_show_left:n { #1 }
3445   \bool_set_true:N \l__enumext_wrap_label_v_bool
3446   \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
3447   \__enumext_keyans_show_item_opt:

```

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

```

3448   \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_store_current_label_tmp_tl
3449   \__enumext_keyans_addto_prop:n { #1 }
3450   \__enumext_keyans_store_ref:
3451   \__enumext_keyans_addto_seq:n { #1 }
3452   \int_gincr:N \g__enumext_check_starred_cmd_int
3453 }

```

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

`\item*`

`__enumext_keyans_redefine_item:`

The function `__enumext_keyans_redefine_item:` is responsible for adding the *starred argument* and *optional argument* by the `__enumext_list_arg_two_v:` function in the definition of the `keyans` environment. Here we need to use `\peek_remove_spaces:n` to prevent an unwanted space when using `\item*` in conjunction with the `itemindent` key. This function are passed to `__enumext_list_arg_two_v:` used in the definition of the `keyans` environment (§13.38).

```

3454 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
3455 {
3456   \RenewDocumentCommand \item { s o }
3457   {
3458     \bool_if:nTF {##1}
3459     {
3460       \peek_remove_spaces:n
3461       {
3462         \__enumext_keyans_starred_item:n {##2}
3463       }
3464     }
3465     {
3466       \__enumext_keyans_default_item:n {##2}
3467     }
3468   }
3469 }

```

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

`__enumext_keyans_make_label:`

`__enumext_keyans_make_label_std:`

`__enumext_keyans_make_label_box:`

The function `__enumext_keyans_make_label:` redefine `\makeLabel` for the keys `mode-box`, `align`, `font`, `wrap-label`, `wrap-label*` 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).

```

3470 \cs_new_protected:Nn \__enumext_keyans_make_label:
3471 {
3472   \IfDocumentMetadataTF
3473   {
3474     \__enumext_keyans_make_label_box:
3475   }
3476   {
3477     \bool_if:NTF \l__enumext_mode_box_bool
3478     {
3479       \__enumext_keyans_make_label_box:
3480     }
3481     {
3482       \__enumext_keyans_make_label_std:

```

```

3483     }
3484   }
3485 }
Standard definition when \DocumentMetadata is not active.
3486 \cs_new_protected:Nn \__enumext_keyans_make_label_std:
3487 {
3488   \RenewDocumentCommand \makeLabel { m }
3489   {
3490     \tl_use:N \l__enumext_label_fill_left_v_tl
3491     \tl_use:N \l__enumext_label_font_style_v_tl
3492     \bool_if:NTF \l__enumext_wrap_label_v_bool
3493     {
3494       \__enumext_wrapper_label_v:n { ##1 }
3495     }
3496     { ##1 }
3497     \tl_use:N \l__enumext_label_fill_right_v_tl
3498   }
3499 }

```

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

```

3500 \cs_new_protected:Nn \__enumext_keyans_make_label_box:
3501 {
3502   \RenewDocumentCommand \makeLabel { m }
3503   {
3504     \strut\smash
3505     {
3506       \makebox[ \l__enumext_labelwidth_v_dim ][ \l__enumext_align_label_pos_v_str ]
3507       {
3508         \tl_use:N \l__enumext_label_font_style_v_tl
3509         \bool_if:NTF \l__enumext_wrap_label_v_bool
3510         {
3511           \__enumext_wrapper_label_v:n { ##1 }
3512         }
3513         { ##1 }
3514       }
3515     }
3516   }
3517 }

```

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

13.38 Second argument of the lists

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

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

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

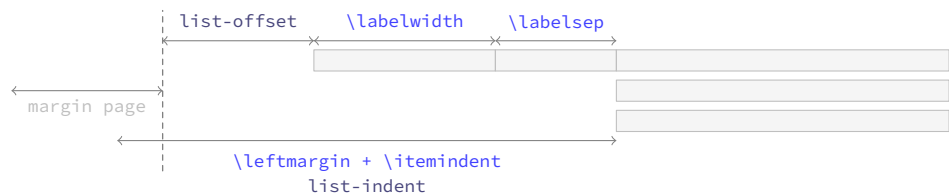


Figure 9: Representation of standard horizontal lengths in `list` environment.

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

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

```

\__enumext_calc_hspace:NNNNNNN
\__enumext_calc_hspace:cccccc

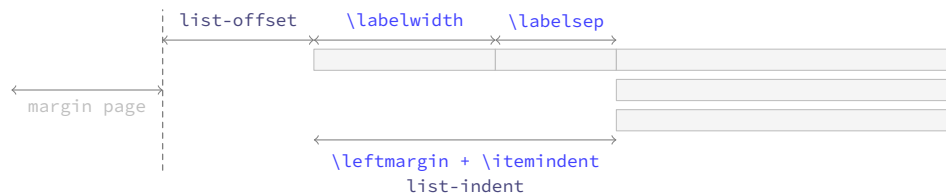
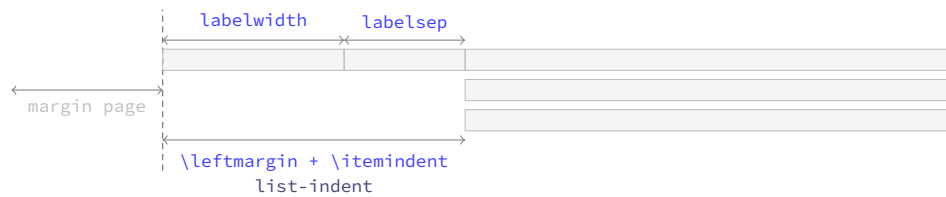
```

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

```

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

```

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

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

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

```

3518 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNN #1 #2 #3 #4 #5 #6 #7
3519 {
3520   \dim_compare:nNnT { #1 } < { \c_zero_dim }
3521   {
3522     \msg_warning:nnnV { enumext } { width-non-positive } { labelwidth } { #1 }
3523     \dim_set:Nn #1 { \dim_abs:n { #1 } }
3524   }
3525   \dim_compare:nNnT { #2 } < { \c_zero_dim }
3526   {
3527     \msg_warning:nnnV { enumext } { width-negative } { labelsep } { #2 }
3528     \dim_set:Nn #2 { \dim_abs:n { #2 } }
3529   }

```

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

```

3530   \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }

```

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

```

3531   \dim_compare:nNnTF { #4 } < { \c_zero_dim }
3532   {
3533     \dim_set:Nn #6 { #1 + #2 - #4 }
3534     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3535   }
3536   {
3537     \dim_compare:nNnT { #4 } = { #1 + #2 }
3538     { \dim_set:Nn #6 { \c_zero_dim } }
3539     \dim_compare:nNnT { #4 } < { #1 + #2 }
3540     { \dim_set:Nn #6 { #1 + #2 - #4 } }
3541     \dim_compare:nNnT { #4 } > { #1 + #2 }
3542     {
3543       \dim_set:Nn #6 { -#1 - #2 + #4 }
3544       \dim_set:Nn #6 { #6*-1 }
3545     }
3546     \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3547   }
3548 }
3549 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { cccccc }

```

(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.

```

\__enumext_list_arg_two_i:
\__enumext_list_arg_two_ii:
\__enumext_list_arg_two_iii:
\__enumext_list_arg_two_iv:
\__enumext_list_arg_two_v:
3550 \cs_set_protected:Npn \__enumext_tmp:n #1
3551 {
3552   \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3553   {
3554     \__enumext_calc_hspace:ccccc
3555     { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }

```

```

3556     { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3557     { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3558     { \__enumext_leftmargin_tmp_#1_bool }
3559 \clist_map_inline:nn
3560   { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3561   { \dim_set_eq:cc {###1} { \__enumext_###1_#1_dim } }
3562 \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3563   { \skip_set_eq:cc {###1} { \__enumext_###1_#1_skip } }
3564 \usecounter { enumX#1 }
3565 \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
3566 \str_if_eq:nnTF {#1} { v }
3567   {
3568     \__enumext_keyans_redefine_item:
3569     \__enumext_keyans_make_label:
3570     \__enumext_keyans_ref:
3571     \__enumext_keyans_fake_item_indent:
3572     \bool_if:cT { \__enumext_show_length_#1_bool }
3573     {
3574       \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3575     }
3576   }
3577   {
3578     \__enumext_redefine_item:
3579     \__enumext_make_label:
3580     \__enumext_standar_ref:
3581     \__enumext_fake_item_indent:
3582     \bool_if:cT { \__enumext_show_length_#1_bool }
3583     {
3584       \msg_term:nnne { enumext } { list-lengths } {#1}
3585       { \int_use:N \__enumext_level_int }
3586     }
3587   }
3588 }
3589 }
3590 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }

```

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

`__enumext_list_arg_two_vii:` For the horizontal environments `enumext*` and `keyans*` the implementation is similar, but, the value of `\partopsep` is always `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 `lisparindent` and `parsep` to set the value of `\parskip` locally.

`__enumext_list_arg_two_viii:`

```

3591 \cs_set_protected:Npn \__enumext_tmp:n #1
3592   {
3593     \cs_new_protected:cpn { \__enumext_list_arg_two_#1: }
3594     {
3595       \bool_set_true:c { \__enumext_leftmargin_tmp_#1_bool }
3596       \dim_zero:c { \__enumext_leftmargin_tmp_#1_dim }
3597       \__enumext_calc_hspace:cccccc
3598       { \__enumext_labelwidth_#1_dim } { \__enumext_labelsep_#1_dim }
3599       { \__enumext_listoffset_#1_dim } { \__enumext_leftmargin_tmp_#1_dim }
3600       { \__enumext_leftmargin_#1_dim } { \__enumext_itemindent_#1_dim }
3601       { \__enumext_leftmargin_tmp_#1_bool }
3602     \clist_map_inline:nn
3603       { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3604       { \dim_set_eq:cc {###1} { \__enumext_###1_#1_dim } }
3605     \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3606       { \skip_set_eq:cc {###1} { \__enumext_###1_#1_skip } }
3607     \skip_set_eq:Nc \parsep { \__enumext_itemsep_#1_skip }
3608     \skip_zero:N \partopsep
3609     \usecounter { enumX#1 }
3610     \setcounter { enumX#1 } { \int_eval:n { \int_use:c { \__enumext_start_#1_int } - 1 } }
3611     \__enumext_starred_ref:
3612     \str_if_eq:nnTF {#1} { vii }
3613     {
3614       \__enumext_fake_item_indent_vii:
3615       \bool_if:cT { \__enumext_show_length_vii_bool }
3616       { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3617     }
3618     {
3619       \__enumext_fake_item_indent_viii:

```



```

3620         \bool_if:cT { \__enumext_show_length_#1_bool }
3621         { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3622     }
3623 }
3624 }
3625 \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 __enumext_level_int to restrict nesting of the environment, set __enumext_standar_bool to “true” and finally call the function __enumext_is_on_first_level: which sets __enumext_standar_first_bool to “true” only if the environment is NOT nested and we are at the “first level”.

```

3626 \cs_new_protected:Nn \__enumext_safe_exec:
3627 {
3628     \__enumext_is_not_nested:
3629     \__enumext_internal_mini_page:
3630     \int_incr:N \__enumext_level_int
3631     \int_compare:nNnT { \__enumext_level_int } > { 4 }
3632     { \msg_fatal:nn { enumext } { list-too-deep } }
3633     \bool_set_true:N \__enumext_standar_bool
3634     \bool_set_false:N \__enumext_starred_bool
3635     \__enumext_is_on_first_level:
3636 }

```

(End of definition for __enumext_safe_exec:.)

__enumext_parse_keys:n The __enumext_parse_store_keys:n function first we will clear the variable __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.

```

3637 \cs_new_protected:Npn \__enumext_parse_keys:n #1
3638 {
3639     \tl_if_novalue:nF {#1}
3640     {
3641         \str_clear:N \__enumext_series_str
3642         \int_compare:nNnTF { \__enumext_level_int } = { 1 }
3643         {
3644             \keys_set:nn { enumext / level-1 } {#1}
3645             \__enumext_parse_series:n {#1}
3646             \__enumext_nested_base_line_fix:
3647         }
3648         {
3649             \exp_args:Ne \keys_set:nn
3650             { enumext / level-\int_use:N \__enumext_level_int } {#1}
3651         }
3652         \__enumext_store_active_keys:n {#1}
3653     }
3654 }

```

(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*`.

```

3655 \cs_new_protected:Nn \__enumext_start_store_level:
3656 {
3657     \bool_lazy_all:nT
3658     {
3659         { \bool_if_p:N \__enumext_store_active_bool }
3660         { \bool_not_p:n { \__enumext_keyans_env_bool } }
3661         { \bool_if_p:N \g__enumext_standar_bool }
3662     }
3663     {
3664         \int_compare:nNnT { \__enumext_level_int } > { 1 }
3665         {

```

```

3666         \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3667         \__enumext_store_level_open:
3668     }
3669 }

```

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

```

3670 \bool_lazy_all:nT
3671 {
3672   { \bool_if_p:N \l__enumext_store_active_bool }
3673   { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3674   { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
3675 }
3676 {
3677   \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3678   {
3679     \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3680     \__enumext_store_level_open:
3681   }
3682 }
3683 }

```

(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*`.

```

3684 \cs_new_protected:Nn \__enumext_stop_store_level:
3685 {
3686   \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
3687   {
3688     \__enumext_store_level_close:
3689   }
3690 }

```

(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.

```

3691 \cs_new_protected:Nn \__enumext_multicols_start:
3692 {
3693   \int_compare:nNnT
3694     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3695     {
3696       \dim_compare:nNnT
3697         { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3698         {
3699           \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
3700             {
3701               ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
3702                 + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
3703                 ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
3704                 - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3705             }
3706         }
3707       \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
3708       \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3709       {
3710         \dim_zero:N \columnseprule
3711       }
3712     }

```

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

```

3712   \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3713   {
3714     \skip_zero:N \multicolsep
3715     \__enumext_multi_addvspace:
3716   }
3717   \raggedcolumns
3718   \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
3719 }
3720 }

```

(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:`.

```

3721 \cs_new_protected:Nn \__enumext_multicols_stop:
3722 {
3723   \int_compare:nNnTF
3724     { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3725     {
3726       \__enumext_stop_list:
3727       \__enumext_stop_store_level:
3728       \end{multicols}
3729       \__enumext_unskip_unkern:
3730       \__enumext_unskip_unkern:
3731       \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
3732     }
3733     {
3734       \__enumext_stop_list:
3735       \__enumext_stop_store_level:
3736     }
3737 }

```

(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.

```

3738 \cs_new_protected:Nn \__enumext_before_list:
3739 {
3740   \__enumext_vspace_above:
3741   \__enumext_before_args_exec:
3742   \__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.

```

3743   \dim_compare:nNnT
3744     { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3745     {
3746       \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
3747       {
3748         \linewidth
3749         - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
3750         - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
3751       }

```

The boolean variable `\l__enumext_minipage_active_X_bool` will be activated and the integer variable `\g__enumext_minipage_stat_int` used by the `\mini-right` 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.

```

3752     \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
3753     \int_gincr:N \g__enumext_minipage_stat_int
3754     \__enumext_minipage_add_space:
3755     \noindent
3756     \__enumext_mini_page{ \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
3757   }
3758   \__enumext_multicols_start:
3759 }

```

(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 `\mini-right` (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.

```

3760 \cs_new_protected:Nn \__enumext_second_part:
3761 {
3762   \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
3763   {
3764     \int_compare:nNt { \g__enumext_minipage_stat_int } = { 1 }
3765     {
3766       \msg_warning:nn { enumext } { missing-miniright }
3767       \miniright
3768     }
3769     \int_gzero:N \g__enumext_minipage_stat_int
3770     \__enumext_unskip_unkern: % remove topsep + [partopsep]
3771     \end__enumext_mini_page
3772   }
3773   {
3774     \__enumext_multicols_stop:
3775   }

```

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.

```

3776   \__enumext_after_stop_list:
3777   \__enumext_check_ans_key_hook:
3778   \__enumext_vspace_below:
3779   \bool_set_false:N \l__enumext_standar_bool
3780   \__enumext_resume_save_counter:
3781 }

```

(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.

```

3782 \cs_new_protected:Nn \__enumext_set_item_width:
3783 {
3784   \dim_set:Nn \itemwidth { \linewidth }
3785   \dim_compare:nT
3786   {
3787     \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim } != \c_zero_dim
3788   }
3789   {
3790     \dim_sub:Nn \itemwidth
3791     {
3792       \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3793     }
3794   }
3795 }

```

(End of definition for `__enumext_set_item_width:`.)

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

```

3796 \NewDocumentEnvironment{enumext}{0} { }
3797 {
3798   \__enumext_safe_exec:
3799   \__enumext_parse_keys:n {#1}
3800   \__enumext_before_list:
3801   \__enumext_start_store_level:
3802   \__enumext_start_list:nn
3803   { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
3804   {
3805     \use:c { __enumext_list_arg_two_ \__enumext_level: : }
3806     \__enumext_before_keys_exec:
3807   }
3808   \__enumext_set_item_width:
3809   \__enumext_after_args_exec:
3810 }
3811 {
3812   \__enumext_second_part:
3813 }

```

(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`.

```
3814 \__enumext_after_env:nn {enumext}
3815   {
3816     \__enumext_execute_after_env:
3817   }
```

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.

```
3818 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3819   {
3820     \bool_if:NF \l__enumext_store_active_bool
3821     {
3822       \msg_error:nnnn { enumext } { wrong-place } { keyans } { save-ans }
3823     }
3824     \int_incr:N \l__enumext_keyans_level_int
3825     \bool_set_true:N \l__enumext_keyans_env_bool
3826     \__enumext_keyans_name_and_start:
3827     % Set false for interfering with enumext nested in keyans (yes, its possible and crazye)
3828     \bool_set_false:N \l__enumext_store_active_bool
3829     \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
3830     {
3831       \msg_error:nn { enumext } { keyans-nested }
3832     }
3833     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3834     {
3835       \msg_error:nn { enumext } { keyans-wrong-level }
3836     }
3837   }
```

(End of definition for `__enumext_keyans_safe_exec:.`)

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

```
3838 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3839   {
3840     \keys_set:nn { enumext / keyans } {#1}
3841   }
```

(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:
3842 \cs_new_protected:Nn \__enumext_before_list_v:
3843   {
3844     \__enumext_vspace_above_v:
3845     \__enumext_before_args_exec_v:
3846     \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3847     {
3848       \dim_set:Nn \l__enumext_minipage_left_v_dim
3849         {
3850           \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3851         }
3852       \bool_set_true:N \l__enumext_minipage_active_v_bool
3853       \int_gincr:N \g__enumext_minipage_stat_int
3854       \__enumext_keyans_minipage_add_space:
3855       \__enumext_mini_page{ \l__enumext_minipage_left_v_dim }
3856     }
3857     \__enumext_keyans_multicols_start:
3858   }
3859 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3860   {
3861     \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3862     {
3863       \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
```

```

3864     {
3865         \dim_set:Nn \l__enumext_columns_sep_v_dim
3866         {
3867             (
3868                 \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3869             ) / \l__enumext_columns_v_int
3870             - \l__enumext_listoffset_v_dim
3871         }
3872     }
3873     \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim
3874     \dim_zero:N \columnseprule % no rule here
3875     \bool_if:NF \l__enumext_minipage_active_v_bool
3876     {
3877         \skip_zero:N \multicolsep
3878         \__enumext_keyans_multi_addvspace:
3879     }
3880     \raggedcolumns
3881     \begin{multicols}{ \l__enumext_columns_v_int }
3882 }
3883 }
3884 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3885 {
3886     \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
3887     {
3888         \__enumext_stop_list:
3889         \end{multicols}
3890         \__enumext_unskip_unkern:
3891         \__enumext_unskip_unkern:
3892         \par\addvspace{ \l__enumext_multicols_below_v_skip }
3893     }
3894     {
3895         \__enumext_stop_list:
3896     }
3897 }
3898 \cs_new_protected:Nn \__enumext_second_part_v:
3899 {
3900     \bool_if:NTF \l__enumext_minipage_active_v_bool
3901     {
3902         \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3903         {
3904             \msg_warning:nn { enumext } { missing-miniright }
3905             \miniright
3906         }
3907         \int_gzero:N \g__enumext_minipage_stat_int
3908         \__enumext_unskip_unkern: % remove \topsep + [\partopsep]
3909         \end__enumext_mini_page
3910         \par\addvspace{ \l__enumext_minipage_after_skip }
3911     }
3912     {
3913         \__enumext_keyans_multicols_stop:
3914     }
3915     \bool_set_false:N \l__enumext_keyans_env_bool
3916     \__enumext_after_stop_list_v:
3917     \__enumext_vspace_below_v:
3918 }

```

(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.

```

3919 \cs_new_protected:Nn \__enumext_keyans_set_item_width:
3920 {
3921     \dim_set:Nn \itemwidth { \linewidth }
3922     \dim_compare:nT
3923     {
3924         \l__enumext_listoffset_v_dim != \c_zero_dim
3925     }
3926     {
3927         \dim_sub:Nn \itemwidth { \l__enumext_listoffset_v_dim }
3928     }
3929 }

```

(End of definition for `__enumext_keyans_set_item_width:`)

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

```

3930 \NewDocumentEnvironment{keyans}{ 0{} }
3931 {
3932   \__enumext_keyans_safe_exec:
3933   \__enumext_keyans_parse_keys:n {#1}
3934   \__enumext_before_list_v:
3935   \__enumext_start_list:nn
3936   { \tl_use:N \l__enumext_label_v_tl }
3937   {
3938     \__enumext_list_arg_two_v:
3939     \__enumext_before_keys_exec_v:
3940   }
3941   \__enumext_keyans_set_item_width:
3942   \__enumext_after_args_exec_v:
3943 }
3944 {
3945   \__enumext_check_starred_cmd:n { item }
3946   \__enumext_second_part_v:
3947 }

```

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

13.41 Tagging PDF support for non-standart list environments

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

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

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

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
3948 \socket_new:nn {tagsupport/__enumext/starred}{ 1 }
3949 \socket_new_plug:nnn {tagsupport/__enumext/starred} {start-list-tags}
3950 {
3951   \tag_resume:n {#1}
3952   \tag_mc_end_push:
3953   \tag_struct_begin:n {tag=LI}
3954   \tag_struct_begin:n {tag=Lbl}
3955   \tag_mc_begin:n {tag=Lbl}
3956 }
3957 \socket_new_plug:nnn {tagsupport/__enumext/starred} {stop-start-tags}
3958 {
3959   \tag_mc_end:
3960   \tag_struct_end:n {tag=Lbl}
3961   \tag_struct_begin:n {tag=LBody}
3962   \tag_struct_begin:n {tag=text-unit}
3963   \tag_struct_begin:n {tag=text}
3964 }
3965 \socket_new_plug:nnn {tagsupport/__enumext/starred} {stop-list-tags}
3966 {
3967   \tag_struct_end:n {tag=text}
3968   \tag_struct_end:n {tag=text-unit}
3969   \tag_struct_end:n {tag=LBody}
3970   \tag_struct_end:n {tag=LI}
3971   \tag_mc_begin_pop:n {}
3972   \tag_suspend:n {#1}
3973 }

```

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

```

3974 \cs_new_protected_nopar:Npn \__enumext_start_list_tag:n #1
3975 {
3976   \IfDocumentMetadataTF
3977   {
3978     \socket_assign_plug:nn {tagsupport/__enumext/starred} {start-list-tags}
3979     \socket_use:nn {tagsupport/__enumext/starred} {#1}

```



```

3980     } {}
3981   }
3982   \cs_new_protected_nopar:Nn \__enumext_stop_start_list_tag:
3983   {
3984     \IfDocumentMetadataTF
3985     {
3986       \socket_assign_plug:nn {tagsupport/__enumext/starred} {stop-start-tags}
3987       \socket_use:nn {tagsupport/__enumext/starred} { }
3988     } {}
3989   }
3990   \cs_new_protected_nopar:Npn \__enumext_stop_list_tag:n #1
3991   {
3992     \IfDocumentMetadataTF
3993     {
3994       \socket_assign_plug:nn {tagsupport/__enumext/starred} {stop-list-tags}
3995       \socket_use:nn {tagsupport/__enumext/starred} {#1}
3996     } {}
3997   }

```

(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 \socket_new:nn {tagsupport/__enumext/keyanspic}{ 0 }
stop-start-tags \socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {start-list-tags}
stop-list-tags  \__enumext_anspic_start_list_tag:
\__enumext_anspic_start_list_tag: {
\__enumext_anspic_stop_start_list_tag: \tag_resume:n {keyanspic}
\__enumext_anspic_stop_list_tag: \tag_mc_end_push:
\tag_struct_begin:n {tag=LI}
\tag_struct_begin:n {tag=Lbl}
\tag_mc_begin:n {tag=Lbl}
}
\socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {stop-start-tags}
{
\tag_mc_end:
\tag_struct_end:n {tag=Lbl}
\tag_struct_begin:n {tag=LBody}
\tag_struct_begin:n {tag=text-unit}
\tag_struct_begin:n {tag=text}
\tag_mc_begin:n {tag=text}
}
\socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {stop-list-tags}
{
\tag_mc_end:
\tag_struct_end:n {tag=text}
\tag_struct_end:n {tag=text-unit}
\tag_struct_end:n {tag=LBody}
\tag_struct_end:n {tag=LI}
\tag_mc_begin_pop:n {}
\tag_suspend:n {keyanspic}
}

```

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

```

4026 \cs_new_protected_nopar:Nn \__enumext_anspic_start_list_tag:
4027 {
4028   \IfDocumentMetadataTF
4029   {
4030     \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {start-list-tags}
4031     \socket_use:n {tagsupport/__enumext/keyanspic}
4032   } {}
4033 }
4034 \cs_new_protected_nopar:Nn \__enumext_anspic_stop_start_list_tag:
4035 {
4036   \IfDocumentMetadataTF
4037   {
4038     \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {stop-start-tags}
4039     \socket_use:n {tagsupport/__enumext/keyanspic}
4040   } {}
4041 }
4042 \cs_new_protected_nopar:Nn \__enumext_anspic_stop_list_tag:
4043 {
4044   \IfDocumentMetadataTF

```

```

4045     {
4046       \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {stop-list-tags}
4047       \socket_use:n {tagsupport/__enumext/keyanspic}
4048     } {}
4049   }

```

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

13.42 The environment `keyanspic` and `\anspic`

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

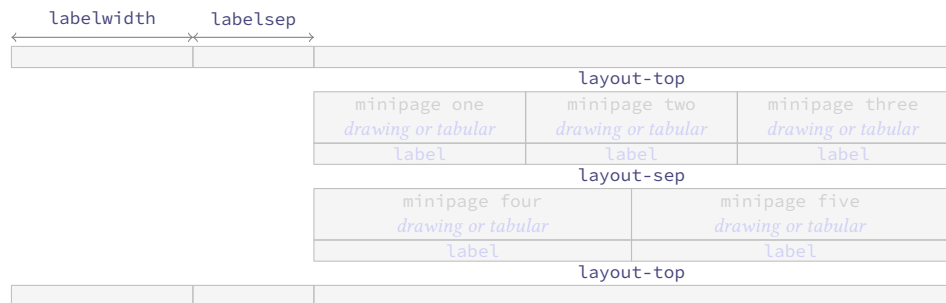


Figure 12: Representation of the `keyanspic` spacing in `enumext`.

In order for the `keyanspic` environment and the `\anspic` command to work correctly, we need to set and export some variables in the first part of the environment definition and pass them to `\anspic` which is executed in the second part of the environment. This implementation is adapted from the answer given by Enrico Gregorio (@egreg) in [How to process the body of an environment and divide it by a \macro?](#).

13.42.1 The environment `keyanspic`

First we define the key that allows us to process the position of the `<label>` centered “above” or “below” which will be `label-pos`, the vertical separation of these from `drawing or tabular` will be handled with the key `label-sep`. The “layout style” will be handled with the key `layout-sty` will take two values separated by comma `{<n° upper, n° lower>}` and will determine the number of `minipage` environments in which all arguments of `\anspic` will be printed at the “upper” and “lower” within the environments separated by the value of the key `layout-sep`. The vertical space “top” and “bottom” of the environment will be handled with the key `layout-top`.

```

4050 \keys_define:nn { enumext / keyanspic }
4051   {
4052     label-pos .choice:,
4053     label-pos / above .code:n =
4054       \bool_set_true:N \l__enumext_anspic_label_above_bool
4055       \str_set:Nn \l__enumext_anspic_mini_pos_str { t },
4056     label-pos / below .code:n =
4057       \bool_set_false:N \l__enumext_anspic_label_above_bool
4058       \str_set:Nn \l__enumext_anspic_mini_pos_str { b },
4059     label-pos / unknown .code:n =
4060       \msg_error:nnee { enumext } { unknown-choice }
4061       { label-pos } { above,~ below } { \exp_not:n {#1} },
4062     label-pos .initial:n = below,
4063     label-pos .value_required:n = true,
4064     label-sep .skip_set:N = \l__enumext_anspic_label_sep_skip,
4065     label-sep .value_required:n = true,
4066     layout-sty .tl_set:N = \l__enumext_anspic_layout_style_tl,
4067     layout-sty .value_required:n = true,
4068     layout-sep .code:n = \keys_set:nn { enumext / keyans }
4069       { parsep = #1 },
4070     layout-sep .value_required:n = true,
4071     layout-top .code:n = \keys_set:nn { enumext / keyans }
4072       { topsep = #1 },
4073     layout-top .value_required:n = true,
4074     unknown .code:n = { \l__enumext_keyans_unknown_keys:n {#1} }
4075   }

```

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

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

`__enumext_keyans_pic_parse_keys:n`

`__enumext_keyans_pic_skip_abs:N`

`\ enumext keyans pic arg two:`

```

4076 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
4077 {
4078   \int_incr:N \l__enumext_keyans_pic_level_int
4079   \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
4080   {
4081     \msg_error:nn { enumext } { keyanspic-nested }
4082   }
4083   \__enumext_keyans_name_and_start:
4084 }

```

Parse [*key = val*] for `keyanspic` environment.

```

4085 \cs_new_protected:Npn \__enumext_keyans_pic_parse_keys:n #1
4086 {
4087   \tl_if_novalue:nF {#1}
4088   {
4089     \keys_set:nn { enumext / keyanspic } {#1}
4090   }
4091 }

```

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

```

4092 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
4093 {
4094   \dim_compare:nNnT { #1 } < { \c_zero_dim }
4095   {
4096     \skip_set:Nn #1 { -#1 }
4097   }
4098 }

```

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.

```

4099 \cs_new_protected:Npn \__enumext_keyans_pic_arg_two:
4100 {
4101   \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
4102   \__enumext_list_arg_two_v:
4103   \__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*.

```

4104   \bool_if:NF \l__enumext_anspic_label_above_bool
4105   {
4106     \stepcounter { enumXv }
4107     \hbox_set:Nn \l__enumext_anspic_label_box { \l__enumext_label_v_tl }
4108     \dim_set:Nn \l__enumext_anspic_label_htdp_dim
4109     {
4110       \box_ht_plus_dp:N \l__enumext_anspic_label_box
4111     }
4112     \skip_add:Nn \parsep
4113     {
4114       \l__enumext_anspic_label_htdp_dim
4115       + \box_dp:N \strutbox
4116       + \l__enumext_anspic_label_sep_skip
4117     }
4118   }

```

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.

```

4119   \dim_add:Nn \leftmargin { -\labelwidth - \labelsep }
4120   \skip_add:Nn \topsep { 0.5\box_dp:N \strutbox }
4121   \dim_zero:N \listparindent
4122   \skip_zero:N \partopsep
4123   \skip_zero:N \itemsep
4124 }

```

(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.

```

4125 \NewDocumentEnvironment{keyanspic}{o}
4126 {
4127   \__enumext_keyans_pic_safe_exec:
4128   \__enumext_keyans_pic_parse_keys:n {#1}
4129   \begin{list} {} { \__enumext_keyans_pic_arg_two: }
4130   \IfDocumentMetadataTF
4131   {
4132     \tag_suspend:n {list}
4133   }{}
4134   \item[] \scan_stop:
4135   \RenewDocumentCommand \item {}
4136   {
4137     \msg_error:nn { enumext } { keyanspic-item-cmd }
4138   }
4139   \IfDocumentMetadataTF
4140   {
4141     \tag_resume:n {keyanspic}
4142     \tag_tool:n {para/tagging=false}
4143     \tag_suspend:n {keyanspic}
4144   } {}
4145 }

```

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.

```

4146 {
4147   \IfDocumentMetadataTF
4148   {
4149     \tag_resume:n {keyanspic}
4150     \tag_mc_end_push:
4151     \tag_struct_begin:n {tag=L,attribute=enumerate}
4152   } {}
4153   \__enumext_anspic_exec:
4154   \IfDocumentMetadataTF
4155   {
4156     \tag_suspend:n {keyanspic}
4157   } {}
4158   \end{list}
4159   \IfDocumentMetadataTF
4160   {
4161     \tag_struct_end:n {tag=L}
4162     \tag_mc_begin_pop:n {}
4163     \tag_struct_end:n {tag=L}
4164     \tag_mc_begin_pop:n {}
4165   } {}

```

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

```

4166   \__enumext_check_starred_cmd:n { anspic }
4167   \setcounter { enumXvi } { 0 }
4168   \bool_if:NTF \l__enumext_anspic_label_above_bool
4169   {
4170     \par\addvspace{ 0.5\box_dp:N \strutbox }
4171   }
4172   {
4173     \par
4174     \addvspace
4175     {
4176       \dim_eval:n
4177       {
4178         \l__enumext_anspic_label_htdp_dim + \box_ht_plus_dp:N \strutbox
4179         + \l__enumext_anspic_label_sep_skip + \l__enumext_topsep_v_skip
4180       }
4181     }
4182   }
4183 }

```

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

13.42.2 The command `\anspic`

The `\anspic` command take three arguments, the *starred versions* `\anspic*[\langle content \rangle]` store the current `\langle label \rangle` 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 `\langle label \rangle`, the `\anspic` command processes the arguments in order, where #1 and #2 correspond to `\langle label \rangle` and #3 to the mandatory argument and puts all this inside a `minipage` environment. If #1 and #2, that is `\langle label \rangle`, 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.

```

\anspic
\__enumext_anspic_body_dim:n
\__enumext_anspic_label:nn
\__enumext_anspic_label_pos:nnn
\__enumext_anspic_args:nnn
\__enumext_anspic_print:n
\__enumext_anspic_print:e
\__enumext_anspic_print:V
\__enumext_anspic_row:n
\__enumext_anspic_exec:
4184 \NewDocumentCommand \anspic { s o +m }
4185 {
4186   \bool_if:NF \__enumext_store_active_bool
4187   {
4188     \msg_error:nnnn { enumext } { wrong-place } { keyanspic } { save-ans }
4189   }
4190   \int_compare:nNt { \__enumext_level_int } > { 1 }
4191   {
4192     \msg_error:nn { enumext } { keyanspic-wrong-level }
4193   }
4194   \int_compare:nNt { \__enumext_keyans_level_int } = { 1 }
4195   {
4196     \msg_error:nnnn { enumext } { command-wrong-place } { anspic } { keyans }
4197   }
4198   \seq_put_right:Nn \__enumext_anspic_args_seq
4199   {
4200     \__enumext_anspic_args:nnn { #1 } { #2 } { #3 }
4201   }
4202 }

```

The `__enumext_anspic_body_dim:n` function will set the value of `__enumext_anspic_body_htdp_dim` equal to the “height plus depth” of the *mandatory argument* if the key `label-pos` is set “below”.

```

4203 \cs_new_protected:Npn \__enumext_anspic_body_dim:n #1
4204 {
4205   \bool_if:NF \__enumext_anspic_label_above_bool
4206   {
4207     \IfDocumentMetadataTF
4208     {
4209       \tag_suspend:n {keyanspic}
4210     } { }
4211     \vbox_set:Nn \__enumext_anspic_body_box { #1 }
4212     \dim_set:Nn \__enumext_anspic_body_htdp_dim
4213     {
4214       \box_ht_plus_dp:N \__enumext_anspic_body_box
4215     }
4216     \IfDocumentMetadataTF
4217     {
4218       \tag_resume:n {keyanspic}
4219     } { }
4220   }
4221 }

```

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 `\langle label \rangle` and *optional argument* in *prop list* and *sequence* and execute the `show-ans`, `show-pos`, `font`, `wrap-label` and `wrap-opt` keys.

```

4222 \cs_new_protected:Npn \__enumext_anspic_label:nn #1 #2
4223 {
4224   \makebox[ \__enumext_anspic_mini_width_dim ][ c ]
4225   {
4226     \bool_if:nT { #1 }
4227     {
4228       \__enumext_keyans_addto_prop:n { #2 }
4229       \__enumext_keyans_store_ref:
4230       \__enumext_keyans_addto_seq:n { #2 }
4231       \int_gincr:N \g__enumext_check_starred_cmd_int
4232       \bool_lazy_or:nnT
4233       { \bool_if_p:N \__enumext_show_answer_bool }

```

```

4234         { \bool_if_p:N \l__enumext_show_position_bool }
4235         {
4236             \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
4237             \__enumext_keyans_show_left:n { #2 }
4238             \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
4239         }
4240     }
4241     \tl_use:N \l__enumext_label_font_style_v_tl
4242     \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl }
4243     \__enumext_keyans_show_item_opt:
4244 }
4245 }

```

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.

```

4246 \cs_new_protected:Npn \__enumext_anspic_label_pos:nnn #1 #2 #3
4247 {
4248     \stepcounter { enumXvi }
4249     \__enumext_anspic_body_dim:n { #3 }
4250     \bool_if:NTF \l__enumext_anspic_label_above_bool
4251     {
4252         \__enumext_anspic_label:nn { #1 } { #2 }
4253     }
4254     {
4255         \raisebox
4256         {
4257             -\dim_eval:n
4258             {
4259                 \l__enumext_anspic_label_htdp_dim
4260                 + \l__enumext_anspic_body_htdp_dim
4261                 + \box_dp:N \strutbox
4262                 + \l__enumext_anspic_label_sep_skip
4263             }
4264         }
4265         [ opt ] [ opt ]
4266         {
4267             \__enumext_anspic_label:nn { #1 } { #2 }
4268         }
4269     }
4270 }
4271 %

```

The `__enumext_anspic_args:nnn` function will be responsible for placing the code compatible with *tagged* PDF and the arguments within the `\l__enumext_anspic_args_seq` sequence which will be processed by the `__enumext_anspic_print:n` function in the second part of the definition of the `keyanspic` environment.

```

4272 \cs_new_protected:Nn \__enumext_anspic_args:nnn
4273 {
4274     \__enumext_anspic_start_list_tag:
4275     \__enumext_anspic_label_pos:nnn { #1 } { #2 } { #3 }
4276     \__enumext_anspic_stop_start_list_tag:
4277     \bool_if:NTF \l__enumext_anspic_label_above_bool
4278     {
4279         \[\[\l__enumext_anspic_label_sep_skip] #3
4280     }
4281     {
4282         \[ #3
4283     }
4284     \__enumext_anspic_stop_list_tag:
4285 }

```

The value `{\langle n° upper, n° 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`.

```

4286 \cs_new_protected:Nn \__enumext_anspic_print:n
4287 {
4288     \clist_map_function:nN { #1 } \__enumext_anspic_row:n
4289 }
4290 \cs_generate_variant:Nn \__enumext_anspic_print:n { e, v }

```

The function `__enumext_anspic_row:n` will set the *widths* for the `minipage` environments and place *all arguments* passed to `\anspic saved` in the `\l__enumext_anspic_args_seq` sequence inside them.

```

4291 \cs_new_protected:Nn \__enumext_anspic_row:n
4292 {

```

```

4293 \dim_set:Nn \l__enumext_anspic_mini_width_dim { \linewidth / #1 }
4294 \int_set:Nn \l__enumext_anspic_above_int { \l__enumext_anspic_below_int }
4295 \int_set:Nn \l__enumext_anspic_below_int { \l__enumext_anspic_above_int + #1 }
4296 \int_step_inline:nnn
4297   { \l__enumext_anspic_above_int + 1 }
4298   { \l__enumext_anspic_below_int }
4299   {
4300     \IfDocumentMetadataTF
4301     {
4302       \tag_suspend:n {minipage}
4303     } { }
4304     \begin{minipage}[ \l__enumext_anspic_mini_pos_str ]{ \l__enumext_anspic_mini_width_dim }
4305       \centering
4306       \seq_item:Nn \l__enumext_anspic_args_seq { ##1 }
4307     \end{minipage}
4308     \IfDocumentMetadataTF
4309     {
4310       \tag_resume:n {minipage}
4311     } { }
4312   }
4313 \par
4314 }

```

The `\l__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*.

```

4315 \cs_new_protected:Nn \l__enumext_anspic_exec:
4316 {
4317   \tl_if_empty:NTF \l__enumext_anspic_layout_style_tl
4318   {
4319     \l__enumext_anspic_print:e { \seq_count:N \l__enumext_anspic_args_seq }
4320   }
4321   {
4322     \l__enumext_anspic_print:v \l__enumext_anspic_layout_style_tl
4323   }
4324 }

```

(End of definition for `\anspic` and others. This function is documented on page 16.)

13.43 The horizontal environments

Generating *horizontal list environments* is NOT as simple as standard \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 $\langle content \rangle$ of this in *horizontal box* using `\makebox` for the `label` and a `minipage` environment for the $\langle content \rangle$ passed to `\item`, we will also add the *optional argument* ($\langle number \rangle$) to `\item` to be able to *join columns* horizontally, in simple terms, we want `\item` to behave in the same way as in the `enumext` environment but adding an *first optional argument* ($\langle number \rangle$).

A side effect is the limitation of using `\item` in this way *without* using `\RenewDocumentCommand`, which loses the original definition and affects the *standard list environments* provided by \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

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

```

\l__enumext_starred_columns_set_viii:
4325 \cs_new_protected:Nn \l__enumext_starred_columns_set_vii:
4326 {
4327   \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
4328   {
4329     \dim_set:Nn \l__enumext_columns_sep_vii_dim
4330     {

```



```

4331         ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
4332         / \l__enumext_columns_vii_int
4333     }
4334 }
4335 \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - 1 }
4336 \dim_set:Nn \l__enumext_item_width_vii_dim
4337 {
4338     ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
4339     / \l__enumext_columns_vii_int
4340     - \l__enumext_labelwidth_vii_dim
4341     - \l__enumext_labelsep_vii_dim
4342 }

```

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

```

4343 \dim_compare:nNnT { \l__enumext_rightmargin_vii_dim } > { \c_zero_dim }
4344 {
4345     \dim_sub:Nn \l__enumext_item_width_vii_dim
4346     {
4347         ( \l__enumext_rightmargin_vii_dim * \l__enumext_tmpa_vii_int )
4348         / \l__enumext_columns_vii_int
4349     }
4350     \dim_add:Nn \l__enumext_columns_sep_vii_dim
4351     {
4352         \l__enumext_rightmargin_vii_dim
4353     }
4354 }
4355 }

```

Same implementation for the `keyans*` environment.

```

4356 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
4357 {
4358     \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
4359     {
4360         \dim_set:Nn \l__enumext_columns_sep_viii_dim
4361         {
4362             ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
4363             / \l__enumext_columns_viii_int
4364         }
4365     }
4366     \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - 1 }
4367     \dim_set:Nn \l__enumext_item_width_viii_dim
4368     {
4369         ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
4370         / \l__enumext_columns_viii_int
4371         - \l__enumext_labelwidth_viii_dim
4372         - \l__enumext_labelsep_viii_dim
4373     }
4374     \dim_compare:nNnT { \l__enumext_rightmargin_viii_dim } > { \c_zero_dim }
4375     {
4376         \dim_sub:Nn \l__enumext_item_width_viii_dim
4377         {
4378             ( \l__enumext_rightmargin_viii_dim * \l__enumext_tmpa_viii_int )
4379             / \l__enumext_columns_viii_int
4380         }
4381         \dim_add:Nn \l__enumext_columns_sep_viii_dim
4382         {
4383             \l__enumext_rightmargin_viii_dim
4384         }
4385     }
4386 }

```

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

13.43.2 Functions for join item columns

```

\__enumext_starred_joined_item_vii:n
\__enumext_starred_joined_item_viii:n

```

The functions `__enumext_starred_joined_item_vii:n` and `__enumext_starred_joined_item_viii:n` will set the *width* of the box in which the `<content>` passed to `\item(<columns>)` will be stored together with the value of `\itemwidth` for the `enumext*` environment.

```

4387 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
4388 {
4389     \int_set:Nn \l__enumext_joined_item_vii_int {#1}
4390     \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
4391     {

```

```

4392     \msg_warning:nnee { enumext } { item-joined }
4393     { \int_use:N \l__enumext_joined_item_vii_int }
4394     { \int_use:N \l__enumext_columns_vii_int }
4395     \int_set:Nn \l__enumext_joined_item_vii_int
4396     {
4397         \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4398     }
4399 }
4400 \int_compare:nNnT
4401 { \l__enumext_joined_item_vii_int }
4402 >
4403 { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4404 {
4405     \msg_warning:nnee { enumext } { item-joined-columns }
4406     { \int_use:N \l__enumext_joined_item_vii_int }
4407     {
4408         \int_eval:n
4409         { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4410     }
4411     \int_set:Nn \l__enumext_joined_item_vii_int
4412     {
4413         \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4414     }
4415 }
4416 \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { 1 }
4417 {
4418     \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
4419     \int_decr:N \l__enumext_joined_item_aux_vii_int
4420     \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
4421     \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
4422     \dim_set:Nn \l__enumext_joined_width_vii_dim
4423     {
4424         \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
4425         + ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
4426           + \l__enumext_columns_sep_vii_dim
4427           ) * \l__enumext_joined_item_aux_vii_int
4428     }
4429     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
4430 }
4431 {
4432     \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
4433     \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
4434 }
4435 }

```

Same implementation for the `keyans*` environment.

```

4436 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
4437 {
4438     \int_set:Nn \l__enumext_joined_item_viii_int {#1}
4439     \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
4440     {
4441         \msg_warning:nnee { enumext } { item-joined }
4442         { \int_use:N \l__enumext_joined_item_viii_int }
4443         { \int_use:N \l__enumext_columns_viii_int }
4444         \int_set:Nn \l__enumext_joined_item_viii_int
4445         {
4446             \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4447         }
4448     }
4449     \int_compare:nNnT
4450     { \l__enumext_joined_item_viii_int }
4451     >
4452     { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4453     {
4454         \msg_warning:nnee { enumext } { item-joined-columns }
4455         { \int_use:N \l__enumext_joined_item_viii_int }
4456         {
4457             \int_eval:n
4458             { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4459         }
4460         \int_set:Nn \l__enumext_joined_item_viii_int
4461         {

```

```

4462         \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4463     }
4464 }
4465 \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { 1 }
4466 {
4467     \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
4468     \int_decr:N \l__enumext_joined_item_aux_viii_int
4469     \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
4470     \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
4471     \dim_set:Nn \l__enumext_joined_width_viii_dim
4472     {
4473         \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
4474         + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
4475             + \l__enumext_columns_sep_viii_dim
4476             ) * \l__enumext_joined_item_aux_viii_int
4477     }
4478     \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
4479 }
4480 {
4481     \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
4482     \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
4483 }
4484 }

```

(End of definition for `__enumext_starred_joined_item_vii:n` and `__enumext_starred_joined_item_viii:n`)

13.43.3 Functions for `mini-env`, `mini-right` and `mini-right*` keys

`__enumext_start_mini_vii:` 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`.

```

4485 \cs_new_protected:Nn \__enumext_start_mini_vii:
4486 {
4487     \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
4488     {
4489         \dim_set:Nn \l__enumext_minipage_left_vii_dim
4490         {
4491             \linewidth
4492             - \l__enumext_minipage_right_vii_dim
4493             - \l__enumext_minipage_hsep_vii_dim
4494         }
4495         \bool_set_true:N \l__enumext_minipage_active_vii_bool
4496         \dim_gset_eq:NN
4497             \g__enumext_minipage_right_vii_dim
4498             \l__enumext_minipage_right_vii_dim
4499         \__enumext_mini_addvspace_vii:
4500         \nointerlineskip\noindent
4501         \__enumext_mini_page{ \l__enumext_minipage_left_vii_dim }
4502     }
4503 }

```

The function `__enumext_stop_mini_vii:` closes the `__enumext_mini_page` environment on the “left side”, applies `\hfill` and set the variable `\g__enumext_minipage_active_vii_bool` to “true” which will be used in the function `__enumext_after_env:n` to execute the `minipage` on the “right side”. At this point we will execute the `__enumext_stop_list:` and `__enumext_stop_store_level_vii:` functions stopping the `list` environment and the level saving mechanism for storage in *sequence* of the `\anskey` command and `anskey*` environment. This function is passed to the `__enumext_after_list_vii:` function in the second part of the `enumext*` environment definition (§13.44).

```

4504 \cs_new_protected:Nn \__enumext_stop_mini_vii:
4505 {
4506     \bool_if:NTF \l__enumext_minipage_active_vii_bool
4507     {
4508         \__enumext_stop_list:
4509         \__enumext_stop_store_level_vii:
4510         \IfDocumentMetadataTF { \tag_resume:n {enumext*} } { }
4511         \end__enumext_mini_page
4512         \hfill
4513         \bool_gset_true:N \g__enumext_minipage_active_vii_bool
4514     }
4515     {
4516         \__enumext_stop_list:

```

```

4517     \__enumext_stop_store_level_vii:
4518   }
4519 }

```

(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.

```

4520 \__enumext_after_env:nn {enumext*}
4521 {
4522   \bool_if:NT \g__enumext_minipage_active_vii_bool
4523   {
4524     \__enumext_minipage:w [ t ] { \g__enumext_minipage_right_vii_dim }
4525     \legacy_if_gset_false:n { @minipage }
4526     \skip_vertical:N \c_zero_skip
4527     \par\addvspace { \g__enumext_minipage_right_skip }
4528     \bool_if:NF \g__enumext_minipage_center_vii_bool
4529     {
4530       \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
4531       {
4532         \centering
4533       }
4534     }
4535     \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
4536     {
4537       \tl_use:N \g__enumext_miniright_code_vii_tl
4538     }
4539     \box_use_drop:N \l__enumext_miniright_code_vii_box
4540     \skip_vertical:N \c_zero_skip
4541     \__enumext_endminipage:
4542     \par\addvspace{ \g__enumext_minipage_after_skip }
4543   }
4544   \bool_gset_false:N \g__enumext_minipage_active_vii_bool
4545   \bool_gset_true:N \g__enumext_minipage_center_vii_bool
4546   \tl_gclear:N \g__enumext_miniright_code_vii_tl
4547   \dim_gzero:N \g__enumext_minipage_right_vii_dim
4548   \bool_gset_false:N \g__enumext_starred_bool
4549 }

```

`__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.

```

4550 \cs_new_protected:Nn \__enumext_start_mini_viii:
4551 {
4552   \dim_compare:nNNT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
4553   {
4554     \dim_set:Nn \l__enumext_minipage_left_viii_dim
4555     {
4556       \linewidth
4557       - \l__enumext_minipage_right_viii_dim
4558       - \l__enumext_minipage_hsep_viii_dim
4559     }
4560     \bool_set_true:N \l__enumext_minipage_active_viii_bool
4561     \dim_gset_eq:NN
4562     \g__enumext_minipage_right_viii_dim
4563     \l__enumext_minipage_right_viii_dim
4564     \__enumext_mini_addvspace_viii:
4565     \nointerlineskip\noindent
4566     \__enumext_mini_page{ \l__enumext_minipage_left_viii_dim }
4567   }
4568 }
4569 \cs_new_protected:Nn \__enumext_stop_mini_viii:
4570 {
4571   \bool_if:NTF \l__enumext_minipage_active_viii_bool
4572   {
4573     \__enumext_stop_list:
4574     \IfDocumentMetadataTF { \tag_resume:n {keyans*} } { }
4575     \end__enumext_mini_page
4576     \hfill
4577     \bool_gset_true:N \g__enumext_minipage_active_viii_bool

```

```

4578     }
4579     {
4580     \__enumext_stop_list:
4581     }
4582   }
4583 \__enumext_after_env:nn {keyans*}
4584 {
4585   \bool_if:NT \g__enumext_minipage_active_viii_bool
4586   {
4587     \__enumext_mini_page{ \g__enumext_minipage_right_viii_dim }
4588     \par\addvspace { \g__enumext_minipage_right_skip }
4589     \bool_if:NF \g__enumext_minipage_center_viii_bool
4590     {
4591       \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
4592       {
4593         \centering
4594       }
4595     }
4596     \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
4597     {
4598       \tl_use:N \g__enumext_miniright_code_viii_tl
4599     }
4600     \box_use_drop:N \l__enumext_miniright_code_viii_box
4601     \end__enumext_mini_page
4602     \par\addvspace{ \g__enumext_minipage_after_skip }
4603   }
4604   \bool_gset_false:N \g__enumext_minipage_active_viii_bool
4605   \bool_gset_true:N \g__enumext_minipage_center_viii_bool
4606   \tl_gclear:N \g__enumext_miniright_code_viii_tl
4607   \dim_gzero:N \g__enumext_minipage_right_viii_dim
4608 }

```

(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.

```

4609 \NewDocumentEnvironment{enumext*}{o}
4610 {
4611   \__enumext_safe_exec_vii:
4612   \__enumext_parse_keys_vii:n {#1}
4613   \__enumext_before_list_vii:
4614   \__enumext_start_store_level_vii:
4615   \__enumext_start_list:nn { }
4616   {
4617     \__enumext_list_arg_two_vii:
4618     \__enumext_before_keys_exec_vii:
4619   }
4620   \IfDocumentMetadataTF { \tag_suspend:n {enumext*} } { }
4621   \__enumext_starred_columns_set_vii:
4622   \item[] \scan_stop:
4623   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_first_item_tmp_vii:
4624   \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
4625   \ignorespaces
4626 }
4627 {
4628   \IfDocumentMetadataTF { \tag_struct_end:n {tag=text-unit} } { }
4629   \__enumext_stop_item_tmp_vii:
4630   \__enumext_remove_extra_parsep_vii:
4631   \__enumext_after_list_vii:
4632 }

```

(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 `__enumext_level_h_int` to restrict nesting of the environment, set `__enumext_starred_bool` to true and finally call the function `__enumext_is_on_first_level`: which sets `__enumext_starred_first_bool` to true if we are not nested, allowing the “*storage system*” to be used.

```

4633 \cs_new_protected:Nn \__enumext_safe_exec_vii:
4634 {
4635   \__enumext_is_not_nested:
4636   \__enumext_internal_mini_page:
4637   \int_incr:N \__enumext_level_h_int
4638   \int_compare:nNnT { \__enumext_level_h_int } > { 1 }
4639   {
4640     \msg_error:nn { enumext } { nested }
4641   }
4642   \int_compare:nNnT { \__enumext_keyans_level_h_int } = { 1 }
4643   {
4644     \msg_error:nnn { enumext } { nested-horizontal } { keyans*}
4645   }
4646   \bool_set_true:N \__enumext_starred_bool
4647   \bool_set_false:N \__enumext_standar_bool
4648   \__enumext_is_on_first_level:
4649 }

```

(End of definition for `__enumext_safe_exec_vii:`)

`__enumext_parse_keys_vii:n` First we will clear the variable `__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.

```

4650 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
4651 {
4652   \tl_if_novalue:nF {#1}
4653   {
4654     \str_clear:N \__enumext_series_str
4655     \keys_set:nn { enumext / enumext* } {#1}
4656     \__enumext_parse_series:n {#1}
4657     \__enumext_store_active_keys_vii:n {#1}
4658   }
4659 }

```

(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*`.

```

4660 \cs_new_protected:Nn \__enumext_before_list_vii:
4661 {
4662   \__enumext_vspace_above_vii:
4663   \__enumext_check_ans_active:
4664   \__enumext_before_args_exec_vii:
4665   \__enumext_start_mini_vii:
4666 }

```

(End of definition for `__enumext_before_list_vii:`)

`__enumext_after_list_vii:` The function `__enumext_after_list_vii:` first calls the function `__enumext_stop_mini_vii:` which internally calls `__enumext_stop_list:` and `__enumext_stop_store_level_vii:` (§13.43.3) used by the keys `mini-env`, `mini-right` and `mini-right*`, then to the functions `__enumext_after_stop_list_vii:` used by the key `after`, `__enumext_check_ans_key_hook:` used by the key `check-ans`, `__enumext_vspace_below_vii:` used by the keys `below` and `below*`. Finally set `__enumext_starred_bool` to false and call the `__enumext_resume_save_counter:` function used by the `series`, `resume` and `resume*` keys.

```

4667 \cs_new_protected:Nn \__enumext_after_list_vii:
4668 {
4669   \__enumext_stop_mini_vii:
4670   \__enumext_after_stop_list_vii:
4671   \__enumext_check_ans_key_hook:
4672   \__enumext_vspace_below_vii:
4673   \bool_set_false:N \__enumext_starred_bool
4674   \__enumext_resume_save_counter:
4675 }

```

(End of definition for `__enumext_after_list_vii:`)

`__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`.

```

4676 \cs_new_protected:Nn \__enumext_start_store_level_vii:
4677 {
4678   \bool_if:NT \l__enumext_store_active_bool
4679   {
4680     \int_compare:nNtT { \l__enumext_level_int } > { 0 }
4681     {
4682       \__enumext_store_level_open_vii:
4683     }
4684   }
4685 }
4686 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
4687 {
4688   \bool_if:NT \l__enumext_store_active_bool
4689   {
4690     \int_compare:nNtT { \l__enumext_level_int } > { 0 }
4691     {
4692       \__enumext_store_level_close_vii:
4693     }
4694   }
4695 }

```

(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.

```

4696 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_vii:
4697 {
4698   \skip_horizontal:n
4699   {
4700     -\l__enumext_labelwidth_vii_dim - \l__enumext_labelsep_vii_dim
4701   }
4702   \ignorespaces
4703 }

```

(End of definition for `__enumext_first_item_tmp_vii:`)

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

```

4704 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
4705 {
4706   \__enumext_stop_item_tmp_vii:
4707   \int_incr:N \l__enumext_item_column_pos_vii_int
4708   \int_gincr:N \g__enumext_item_count_all_vii_int
4709   \__enumext_item_peek_args_vii:
4710 }

```

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).

```

4711 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
4712 {
4713   \peek_meaning:NTF (
4714     { \__enumext_joined_item_vii:w }
4715     { \__enumext_joined_item_vii:w (1) }
4716   }

```

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`.

```

4717 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)

```



```

4718 {
4719   \__enumext_starred_joined_item_vii:n {#1}
4720   \peek_meaning_remove:NTF *
4721   { \__enumext_starred_item_vii:w }
4722   { \__enumext_standar_item_vii:w }
4723 }

```

The function `__enumext_standar_item_vii:w` will first look for the argument “[”, if present it will set the state of the variable `__enumext_wrap_label_opt_vii_bool` equal to the state of the variable `__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 `__enumext_wrap_label_vii_bool` handled by the `wrap-label` key to true and set the switch `\if@noitemarg` to true to execute the enumerated version of `\item` by means of the function `__enumext_start_item_vii:w [__enumext_label_vii_tl]`.

```

4724 \cs_new_protected:Npn \__enumext_standar_item_vii:w
4725 {
4726   \bool_set_false:N \__enumext_item_starred_vii_bool
4727   \peek_meaning:NTF [
4728   {
4729     \bool_set_eq:NN \__enumext_wrap_label_vii_bool \__enumext_wrap_label_opt_vii_bool
4730     \__enumext_start_item_vii:w
4731   }
4732   {
4733     \bool_set_true:N \__enumext_wrap_label_vii_bool
4734     \legacy_if_set_true:n { @noitemarg }
4735     \__enumext_start_item_vii:w [ \__enumext_label_vii_tl ] \ignorespaces
4736   }
4737 }

```

The function `__enumext_starred_item_vii:w` together with the specified auxiliary functions `aux_i:w`, `aux_ii:w`, and `aux_iii:w` execute `\item*`, `\item*[⟨symbol⟩]` and `\item*[⟨symbol⟩][⟨offset⟩]`.

```

4738 \cs_new_protected:Npn \__enumext_starred_item_vii:w
4739 {
4740   \bool_set_true:N \__enumext_item_starred_vii_bool
4741   \bool_set_true:N \__enumext_wrap_label_vii_bool
4742   \peek_meaning:NTF [
4743   { \__enumext_starred_item_vii_aux_i:w }
4744   { \__enumext_starred_item_vii_aux_ii:w }
4745 }
4746 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
4747 {
4748   \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
4749   \__enumext_starred_item_vii_aux_ii:w
4750 }
4751 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
4752 {
4753   \peek_meaning:NTF [
4754   { \__enumext_starred_item_vii_aux_iii:w }
4755   {
4756     \dim_set_eq:NN \__enumext_item_symbol_sep_vii_dim \__enumext_labelsep_vii_dim
4757     \legacy_if_set_true:n { @noitemarg }
4758     \__enumext_start_item_vii:w [ \__enumext_label_vii_tl ] \ignorespaces
4759   }
4760 }
4761 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
4762 {
4763   \dim_set:Nn \__enumext_item_symbol_sep_vii_dim {#1}
4764   \legacy_if_set_true:n { @noitemarg }
4765   \__enumext_start_item_vii:w [ \__enumext_label_vii_tl ] \ignorespaces
4766 }

```

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

`__enumext_fake_make_label_vii:n`

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

- ◆ For compatibility with *tagged* PDF and *hyperref* when an environment `enumext` is nested in `enumext*` and the key `save-ans` is not active need setting the `\if@hyper@item` switch to “true”. The explanation for this is given by the master Heiko Oberdiek on `\refstepcounter{enumi}` twice (or more) creates destination with the same identifier. This patch is only needed if you are running `pdflatex` and not if you are running `lualatex`

```

4767 \cs_new_protected_nopar:Npn \__enumext_fake_make_label_vii:n #1
4768 {
4769   \legacy_if:nT { @noitemarg }
4770   {
4771     \legacy_if_set_false:n { @noitemarg }
4772     \legacy_if:nT { @nmbrrlist }
4773     {
4774       \IfDocumentMetadataTF
4775       {
4776         \bool_if:NT \l__enumext_hyperref_bool
4777         {
4778           \legacy_if_set_true:n { @hyper@item }
4779         }
4780       } { }
4781       \refstepcounter{enumXvii}
4782       \bool_if:NT \l__enumext_check_answers_bool
4783       {
4784         \int_gincr:N \g__enumext_item_number_int
4785         \bool_set_true:N \l__enumext_item_number_bool
4786       }
4787     }
4788   }
4789   \bool_if:NT \l__enumext_item_starred_vii_bool
4790   {
4791     \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
4792     {
4793       \tl_gset_eq:NN
4794       \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
4795     }
4796     \mode_leave_vertical:
4797     \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
4798     \hbox_overlap_left:n { \g__enumext_item_symbol_aux_vii_tl }
4799     \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
4800     \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
4801   }
4802   \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
4803   {
4804     \tl_use:N \l__enumext_label_font_style_vii_tl
4805     \bool_if:NTF \l__enumext_wrap_label_vii_bool
4806     {
4807       \__enumext_wrapper_label_vii:n {#1}
4808     }
4809     { #1 }
4810   }
4811   \skip_horizontal:N \l__enumext_labelsep_vii_dim \ignorespaces
4812 }

```

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

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

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

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

```

4813 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
4814 {
4815   \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
4816   \hbox_set_to_wd:Nnw \l__enumext_item_text_vii_box
4817   {
4818     \l__enumext_joined_width_vii_dim
4819     + \l__enumext_labelwidth_vii_dim
4820     + \l__enumext_labelsep_vii_dim
4821   }

```

Redefine the `\footnote` command.

```
4822     \__enumext_renew_footnote_starred:
```

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

```
4823     \__enumext_start_list_tag:n {enumext*}
4824     \__enumext_fake_make_label_vii:n {#1}
4825     \__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.

```
4826     \__enumext_minipage:w [ t ]{ \__enumext_joined_width_vii_dim }
4827     \dim_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
4828     \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
4829     \__enumext_unskip_unkern:
4830     \__enumext_unskip_unkern:
4831     \skip_horizontal:n { -\l__enumext_listparindent_vii_dim } \ignorespaces
4832     \tl_use:N \l__enumext_fake_item_indent_vii_tl
4833     \tl_use:N \l__enumext_after_list_args_vii_tl
4834 }
```

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*.

```
4835 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
4836 {
4837     \__enumext_endminipage:
4838     \__enumext_stop_list_tag:n {enumext*}
4839     \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*.

```
4840     \int_set:Nn \hbadness { 10000 }
4841     \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`.

```
4842     \int_compare:nNnTF
4843     { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
4844     {
4845         \par\noindent
4846         \int_zero:N \l__enumext_item_column_pos_vii_int
4847     }
4848     {
4849         \skip_horizontal:N \l__enumext_columns_sep_vii_dim
4850     }
4851 }
```

(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 `\removelastskip` 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*).

```
4852 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
4853 {
4854     \int_compare:nNnT
4855     {
4856         \int_mod:nn
4857         { \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
4858     }
4859     =
4860     { 0 }
4861     {
4862         \para_end:
4863         \skip_vertical:n { -\l__enumext_itemsep_vii_skip }
4864         \skip_vertical:N \c_zero_skip
4865         \int_gzero:N \g__enumext_item_count_all_vii_int
4866     }
4867 }
```

(End of definition for `__enumext_remove_extra_parsep_viii:`)

As we don't want our check to be executed `check-ans` by levels but on the complete list, we will take it out of the `enumext*` environment using the "hook" function `__enumext_after_env:nn`.

```
4868 \__enumext_after_env:nn {enumext*}
4869   {
4870     \__enumext_execute_after_env:
4871   }
```

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.

```
4872 \NewDocumentEnvironment{keyans*}{o }
4873   {
4874     \__enumext_safe_exec_viii:
4875     \__enumext_parse_keys_viii:n {#1}
4876     \__enumext_before_list_viii:
4877     \__enumext_start_list:nn { }
4878     {
4879       \__enumext_list_arg_two_viii:
4880       \__enumext_before_keys_exec_viii:
4881     }
4882     \IfDocumentMetadataTF { \tag_suspend:n {keyans*} } { }
4883     \__enumext_starred_columns_set_viii:
4884     \item[] \scan_stop:
4885     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_first_item_tmp_viii:
4886     \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
4887     \ignorespaces
4888   }
4889   {
4890     \IfDocumentMetadataTF { \tag_struct_end:n {tag=text-unit} } { }
4891     \__enumext_stop_item_tmp_viii:
4892     \__enumext_remove_extra_parsep_viii:
4893     \__enumext_check_starred_cmd:n { item }
4894     \__enumext_after_list_viii:
4895   }
```

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

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

```
4896 \cs_new_protected:Nn \__enumext_safe_exec_viii:
4897   {
4898     \bool_if:NF \l__enumext_store_active_bool
4899     {
4900       \msg_error:nnnn { enumext } { wrong-place } { keyans* } { save-ans }
4901     }
4902     \int_incr:N \l__enumext_keyans_level_h_int
4903     \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
4904     {
4905       \msg_error:nn { enumext } { nested }
4906     }
4907     \__enumext_keyans_name_and_start:
4908     \bool_if:NT \l__enumext_starred_bool
4909     {
4910       \msg_error:nnn { enumext } { nested-horizontal } { enumext* }
4911     }
4912     \bool_set_true:N \l__enumext_starred_bool
4913     % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
4914     \bool_set_false:N \l__enumext_store_active_bool
4915     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
4916     {
4917       \msg_error:nn { enumext } { keyans-wrong-level }
4918     }
4919   }
```

(End of definition for `__enumext_safe_exec_viii:`)

```

\__enumext_parse_keys_viii:n Parse [key = val] for keyans*.
4920 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
4921 {
4922   \tl_if_novalue:nF {#1}
4923   {
4924     \keys_set:nn { enumext / keyans* } {#1}
4925   }
4926 }

```

(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`.

```

4927 \cs_new_protected:Nn \__enumext_before_list_viii:
4928 {
4929   \__enumext_vspace_above_viii:
4930   \__enumext_before_args_exec_viii:
4931   \__enumext_start_mini_viii:
4932 }

```

(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.

```

4933 \cs_new_protected:Nn \__enumext_after_list_viii:
4934 {
4935   \__enumext_stop_mini_viii:
4936   \__enumext_after_stop_list_viii:
4937   \__enumext_vspace_below_viii:
4938 }

```

(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.

```

4939 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_viii:
4940 {
4941   \skip_horizontal:n
4942   {
4943     -\__enumext_labelwidth_viii_dim - \__enumext_labelsep_viii_dim
4944   }
4945   \ignorespaces
4946 }

```

(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
`__enumext_item_peek_args_viii:` increment the value of `__enumext_item_column_pos_viii_int` that will count the item’s by rows and
`__enumext_joined_item_viii:w` the value of `\g__enumext_item_count_all_viii_int` that will count the total of item’s in the environment.
`__enumext_standar_item_viii:w` After that we will call the function `__enumext_item_peek_args_viii:` that will handle the arguments passed to `\item`.

```

4947 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
4948 {
4949   \__enumext_stop_item_tmp_viii:
4950   \int_incr:N \__enumext_item_column_pos_viii_int
4951   \int_gincr:N \g__enumext_item_count_all_viii_int
4952   \__enumext_item_peek_args_viii:
4953 }

```

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).

```
4954 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
4955   {
4956     \peek_meaning:NTF (
4957       { \__enumext_joined_item_viii:w }
4958       { \__enumext_joined_item_viii:w (1) }
4959   }
```

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`.

```
4960 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
4961   {
4962     \__enumext_starred_joined_item_viii:n {#1}
4963     \peek_meaning_remove:NTF *
4964     { \__enumext_starred_item_viii:w }
4965     { \__enumext_standar_item_viii:w }
4966   }
```

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]`.

```
4967 \cs_new_protected:Npn \__enumext_standar_item_viii:w
4968   {
4969     \bool_set_false:N \l__enumext_item_starred_viii_bool
4970     \peek_meaning:NTF [
4971       {
4972         \bool_set_eq:NN \l__enumext_wrap_label_viii_bool \l__enumext_wrap_label_opt_viii_bool
4973         \__enumext_start_item_viii:w
4974       }
4975       {
4976         \bool_set_true:N \l__enumext_wrap_label_viii_bool
4977         \legacy_if_set_true:n { @noitemarg }
4978         \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ] \ignorespaces
4979       }
4980   }
```

(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_starred_item_exec:
```

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>]`.

```
4981 \cs_new_protected:Npn \__enumext_starred_item_viii:w
4982   {
4983     \bool_set_true:N \l__enumext_item_starred_viii_bool
4984     \bool_set_true:N \l__enumext_wrap_label_viii_bool
4985     \peek_meaning:NTF [
4986       { \__enumext_starred_item_viii_aux_i:w }
4987       { \__enumext_starred_item_viii_aux_ii:w }
4988   }
```

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`.

```
4989 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
4990   {
4991     \tl_clear:N \l__enumext_store_current_label_tl
4992     \tl_if_novalue:nF { #1 }
4993     {
4994       \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
4995       {
4996         \tl_put_right:Ne \l__enumext_store_current_label_tl
4997           {
4998             \l__enumext_store_keyans_item_opt_sep_tl
```

```

4999     }
5000     \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
5001   }
5002   \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
5003 }
5004 \__enumext_starred_item_viii_aux_ii:w
5005 }
5006 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
5007 {
5008   \legacy_if_set_true:n { @noitemarg }
5009   \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ] \ignorespaces
5010 }

```

The function `__enumext_starred_item_exec:` will be in charge of storing the current (*label*) for `\item*` followed by the [*content*] for `\item*[(content)]` if present in the *sequence* and *prop list* set by the `save-ans` key. In this same function the keys `show-ans`, `show-pos` and `save-ref` are implemented.

```

5011 \cs_new_protected:Nn \__enumext_starred_item_exec:
5012 {
5013   \tl_put_left:Ne \l__enumext_store_current_label_tl { \l__enumext_label_viii_tl }
5014   \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
5015   \__enumext_keyans_store_ref:
5016   \tl_put_left:Ne \l__enumext_store_current_label_tl { \item }
5017   \__enumext_keyans_addto_seq_link:
5018   \int_gincr:N \g__enumext_check_starred_cmd_int
5019   \bool_if:NT \l__enumext_show_answer_bool
5020   {
5021     \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
5022   }
5023   \bool_if:NT \l__enumext_show_position_bool
5024   {
5025     \tl_set:Ne \l__enumext_mark_answer_sym_tl
5026     {
5027       \group_begin:
5028         \exp_not:N \normalfont
5029         \exp_not:N \footnotesize [ \int_eval:n
5030           {
5031             \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
5032           }
5033         ]
5034       \group_end:
5035     }
5036     \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
5037   }
5038 }

```

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

`__enumext_fake_make_label_viii:n`

The implementation at this is very similar to that of the `enumext*` environment.

```

5039 \cs_new_protected_nopar:Npn \__enumext_fake_make_label_viii:n #1
5040 {
5041   \legacy_if:nT { @noitemarg }
5042   {
5043     \legacy_if_set_false:n { @noitemarg }
5044     \legacy_if:nT { @nmbrrlist }
5045     {
5046       \refstepcounter{enumXviii}
5047     }
5048   }
5049   \bool_if:NT \l__enumext_item_starred_viii_bool
5050   {
5051     \__enumext_starred_item_exec:
5052   }
5053   \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
5054   {
5055     \tl_use:N \l__enumext_label_font_style_viii_tl
5056     \bool_if:NTF \l__enumext_wrap_label_viii_bool
5057     {
5058       \__enumext_wrapper_label_viii:n {#1}
5059     }
5060     { #1 }
5061   }

```



```

5062   \skip_horizontal:N \l__enumext_labelsep_viii_dim \ignorespaces
5063   }

```

(End of definition for `__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:
5064 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
5065   {
5066     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
5067     \hbox_set_to_wd:Nnw \l__enumext_item_text_viii_box
5068     {
5069       \l__enumext_joined_width_viii_dim
5070       + \l__enumext_labelwidth_viii_dim
5071       + \l__enumext_labelsep_viii_dim
5072     }
5073     \__enumext_renew_footnote_starred:
5074     \__enumext_start_list_tag:n {keyans*}
5075     \__enumext_fake_make_label_viii:n {#1}
5076     \__enumext_stop_start_list_tag:
5077     \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim }
5078     \dim_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
5079     \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
5080     \__enumext_unskip_unkern:
5081     \__enumext_unskip_unkern:
5082     \skip_horizontal:n { -\l__enumext_listparindent_viii_dim } \ignorespaces
5083     \tl_use:N \l__enumext_fake_item_indent_viii_tl
5084     \bool_if:NT \l__enumext_item_starred_viii_bool
5085     {
5086       \__enumext_keyans_show_item_opt:
5087     }
5088     \tl_use:N \l__enumext_after_list_args_viii_tl
5089   }
5090 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
5091   {
5092     \__enumext_endminipage:
5093     \__enumext_stop_list_tag:n {keyans*}
5094     \hbox_set_end:
5095     \int_set:Nn \hbadness { 10000 }
5096     \box_use_drop:N \l__enumext_item_text_viii_box
5097     \int_compare:nNnTF
5098     { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
5099     {
5100       \par\noindent
5101       \int_zero:N \l__enumext_item_column_pos_viii_int
5102     }
5103     {
5104       \skip_horizontal:N \l__enumext_columns_sep_viii_dim
5105     }
5106   }

```

(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.

```

5107 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
5108   {
5109     \int_compare:nNnTF
5110     {
5111       \int_mod:nn
5112       { \g__enumext_item_count_all_viii_int }
5113       { \l__enumext_columns_viii_int }
5114     }
5115     =
5116     { 0 }
5117     {
5118       \para_end:
5119       \skip_vertical:n { -\l__enumext_itemsep_viii_skip }
5120       \skip_vertical:N \c_zero_skip
5121       \int_gzero:N \g__enumext_item_count_all_viii_int
5122     }
5123   }

```

(End of definition for `__enumext_remove_extra_parsep_viii:`)

13.46 The command `\getkeyans`

```
\getkeyans
\__enumext_getkeyans_aux:n
\__enumext_getkeyans:nn
```

The `\getkeyans` command takes a *mandatory argument* of the form `{(store name : position)}`. Retrieve a “single content” stored by `\anskey`, `\anspic*` and `\item*` and `anskey*` from *prop list* defined by `save-ans key`.

```
5124 \NewDocumentCommand \getkeyans { m }
5125   {
5126     \exp_args:Ne \__enumext_getkeyans_aux:n
5127     { \tl_to_str:e { \text_expand:n {#1} } }
5128   }
```

The internal function `__enumext_getkeyans_aux:n` is in charge of *splitting* the *mandatory argument* using “.”. If “.” is omitted it will return an error.

```
5129 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
5130   {
5131     \str_if_in:nnTF {#1} { : }
5132     {
5133       \use:e
5134       {
5135         \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
5136         { {##1} {##2} }
5137       }
5138       \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
5139     }
5140     { \msg_error:nnn { enumext } { missing-colon } {#1} }
5141   }
```

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*.

```
5142 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
5143   {
5144     \prop_if_exist:cTF { g__enumext_#1_prop }
5145     {
5146       \prop_item:cn { g__enumext_#1_prop }{#2}
5147     }
5148     {
5149       \msg_error:nnn { enumext } { undefined-storage-anskey } {#1}
5150     }
5151   }
```

(End of definition for `\getkeyans`, `__enumext_getkeyans_aux:n`, and `__enumext_getkeyans:nn`. This function is documented on page 17.)

13.47 The command `\printkeyans`

The `\printkeyans` command prints “all stored content” in the *sequence* defined by the `save-ans key`. The first thing we will do is define a set of *filtered keys* with which we will control the options of the different nesting levels for the environment `enumext` and `enumext*` by storing their values in the list of tokens `\l__enumext_print_keyans_X_tl`.

The variable `\l__enumext_print_keyans_starred_tl` will have the default *keys* for `\printkeyans*` and will be set by `\setenumext[⟨print*⟩]` and the variable `\l__enumext_print_keyans_vii_tl` will have the default keys for the environment `enumext*` nested within the *sequence* and will be set by `\setenumext[⟨print,*⟩]`, the rest of the variables will be for the environment `enumext` and will be set by `\setenumext[⟨print,level⟩]`.

```
5152 \keys_define:nn { enumext / print }
5153   {
5154     print* .code:n = \keys_precompile:neN { enumext / enumext* }
5155                 { \__enumext_filter_save_key:n {#1} }
5156                 \l__enumext_print_keyans_starred_tl, % starred cmd
5157     print* .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
5158     print-1 .code:n = \keys_precompile:neN { enumext / level-1 }
5159                  { \__enumext_filter_save_key:n {#1} }
5160                  \l__enumext_print_keyans_i_tl,
5161     print-1 .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
5162     print-2 .code:n = \keys_precompile:neN { enumext / level-2 }
5163                  { \__enumext_filter_save_key:n {#1} }
5164                  \l__enumext_print_keyans_ii_tl,
5165     print-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
5166     print-3 .code:n = \keys_precompile:neN { enumext / level-3 }
```

```

5167         { \__enumext_filter_save_key:n {#1} }
5168         \l__enumext_print_keyans_iii_tl,
5169     print-3 .initial:n = { nosep, label=\roman*., first=\small, font=\small },
5170     print-4 .code:n    = \keys_precompile:neN { enumext / level-4 }
5171         { \__enumext_filter_save_key:n {#1} }
5172         \l__enumext_print_keyans_iv_tl,
5173     print-4 .initial:n = { nosep, label=\Alph*., first=\small, font=\small },
5174     print-* .code:n    = \keys_precompile:neN { enumext / enumext* }
5175         { \__enumext_filter_save_key:n {#1} }
5176         \l__enumext_print_keyans_vii_tl, % starred nested
5177     print-* .initial:n = { nosep, label=\arabic*., first=\small, font=\small },
5178 }

```

- The reason for storing *(keys)* in token lists using `\keys_precompile:neN` is because the keys are set via `\setenumext` but are later executed by running the command `\printkeyans` and they are not handled directly by its *optional argument*, except those related to the *first* opening level.

`\printkeyans`

`__enumext_printkeyans:nnn`

Create a user command to print “*all stored content*” in *sequence* for `\anskey`, `anskey*`, `\item*` and `\anspic*`. Within a group we will run our “*precompiled keys*” and then call the internal function `__enumext_printkeyans:nnn`.

```

5179 \NewDocumentCommand \printkeyans { s O{ } m }
5180 {
5181     \group_begin:
5182         \tl_use:N \l__enumext_print_keyans_i_tl
5183         \tl_use:N \l__enumext_print_keyans_ii_tl
5184         \tl_use:N \l__enumext_print_keyans_iii_tl
5185         \tl_use:N \l__enumext_print_keyans_iv_tl
5186         \tl_use:N \l__enumext_print_keyans_vii_tl
5187         \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
5188     \group_end:
5189 }

```

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.

```

5190 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
5191 {
5192     \seq_if_exist:cTF { g__enumext_#3_seq }
5193     {
5194         \seq_if_empty:cF { g__enumext_#3_seq }
5195         {

```

If the *starred argument* `*` is present we will check that the environment `enumext*` is not saved in the *sequence*, then execute the variable `\l__enumext_print_keyans_starred_tl` that contains the default *(keys)* for the environment `enumext*`, we set `\l__enumext_base_line_fix_bool` and `\l__enumext_print_keyans_star_bool` to true for *baseline correction*, open the `enumext*` environment passing the *optional argument* and map the *sequence*, then set `\l__enumext_base_line_fix_bool` and `\l__enumext_print_keyans_star_bool` to false.

```

5196         \bool_if:nTF {#1}
5197         {
5198             \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
5199             {
5200                 \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
5201             }
5202             {
5203                 \tl_use:N \l__enumext_print_keyans_starred_tl
5204                 \bool_set_true:N \l__enumext_base_line_fix_bool
5205                 \bool_set_true:N \l__enumext_print_keyans_star_bool
5206                 \begin{enumext*}[#2]
5207                     \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5208                     \end{enumext*}
5209                 \bool_set_false:N \l__enumext_base_line_fix_bool
5210                 \bool_set_false:N \l__enumext_print_keyans_star_bool
5211             }
5212         }

```

Otherwise it will open the environment `enumext` passing the *optional argument* to the “*first level*” then map the *sequence*.

```

5213         {
5214             \begin{enumext}[#2]
5215                 \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5216             \end{enumext}
5217         }

```

```

5218     }
5219   }
5220   {
5221     \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
5222   }
5223 }

```

(End of definition for `\printkeyans` and `__enumext_printkeyans:nnn`. This function is documented on page 18.)

13.48 The command `\setenumext`

The command `\setenumext` will be in charge of managing the `<keys>` passed to all environments and to the `\printkeyans` command. We must take precautions with the `enumext*` environment and “first level” of the `enumext` environment so as not to capture `<keys>` that complicate us.

The function `__enumext_filter_first_level:n` will be in charge of filtering the `<keys>` passed to the environment `enumext*` and “first level” of the environment `enumext`.

```

\__enumext_filter_first_level:n
\__enumext_filter_first_level_key:n
\__enumext_filter_first_level_pair:nn
5224 \cs_new:Npn \__enumext_filter_first_level:n #1
5225 {
5226   \use:e
5227   {
5228     \keyval_parse:NNn
5229     \__enumext_filter_first_level_key:n
5230     \__enumext_filter_first_level_pair:nn {#1}
5231   }
5232 }

```

The function `__enumext_filter_first_level_key:n` will be responsible for filtering the `<keys>` that are passed “without value” by excluding the keys `resume` and `resume*`.

```

5233 \cs_new:Npn \__enumext_filter_first_level_key:n #1
5234 {
5235   \str_case:nnF {#1}
5236   {
5237     { resume } {}
5238     { resume* } {}
5239   }
5240   { , { \exp_not:n {#1} } }
5241 }

```

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.

```

5242 \cs_new:Npn \__enumext_filter_first_level_pair:nn #1#2
5243 {
5244   \str_case:nnF {#1}
5245   {
5246     { series } {}
5247     { resume } {}
5248     { save-ans } {}
5249   }
5250   { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
5251 }

```

(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`.

```

5252 \keys_define:nn { enumext / meta-families }
5253 {
5254   enumext-1 .code:n =
5255     {
5256       \keys_set:ne { enumext / level-1 }
5257       {
5258         \__enumext_filter_first_level:n {#1}
5259       }
5260     } ,
5261   enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
5262   enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
5263   enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
5264   keyans .code:n = { \keys_set:nn { enumext / keyans } {#1} } ,
5265   enumext* .code:n =
5266     {
5267       \keys_set:ne { enumext / enumext* }
5268     }

```

```

5269         \__enumext_filter_first_level:n {#1}
5270     }
5271     } ,
5272     keyans* .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
5273     print* .code:n = { \keys_set:nn { enumext / print } { print* = {#1} } } ,
5274     print-1 .code:n = { \keys_set:nn { enumext / print } { print-1 = {#1} } } ,
5275     print-2 .code:n = { \keys_set:nn { enumext / print } { print-2 = {#1} } } ,
5276     print-3 .code:n = { \keys_set:nn { enumext / print } { print-3 = {#1} } } ,
5277     print-4 .code:n = { \keys_set:nn { enumext / print } { print-4 = {#1} } } ,
5278     print-* .code:n = { \keys_set:nn { enumext / print } { print-* = {#1} } } ,
5279     unknown .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
5280 }

```

We store them in the constant sequence `__enumext_all_families_seq` separated by commas.

```

5281 \seq_const_from_clist:Nn \__enumext_all_families_seq
5282 {
5283     enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
5284     keyans*, print-1, print-2, print-3, print-4, print-*, print*,
5285 }

```

Now we define the user command `\setenumext`.

```

\__enumext_set_parse:n
\__enumext_set_error:nn
5286 \NewDocumentCommand \setenumext { 0{enumext,1} +m }
5287 {
5288     \seq_clear:N \__enumext_setkey_tmpa_seq
5289     \seq_set_from_clist:Nn \__enumext_setkey_tmpb_seq {#1}
5290     \int_set:Nn \__enumext_setkey_tmpa_int
5291     {
5292         \seq_count:N \__enumext_setkey_tmpb_seq
5293     }
5294     \int_compare:nNnTF { \__enumext_setkey_tmpa_int } > { 1 }
5295     {
5296         \seq_pop_left:NN \__enumext_setkey_tmpb_seq \__enumext_setkey_tmpa_tl
5297         \seq_map_function:NN \__enumext_setkey_tmpb_seq \__enumext_set_parse:n
5298         \seq_set_map_e:NNn \__enumext_setkey_tmpa_seq \__enumext_setkey_tmpa_seq
5299         {
5300             \tl_use:N \__enumext_setkey_tmpa_tl - ##1
5301         }
5302     }
5303     {
5304         \seq_put_right:Ne \__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
5305     }
5306     \seq_if_empty:NNTF \__enumext_setkey_tmpa_seq
5307     { \seq_map_inline:Nn \__enumext_all_families_seq }
5308     { \seq_map_inline:Nn \__enumext_setkey_tmpa_seq }
5309     {
5310         \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
5311     }
5312 }

```

Internal functions used by the `\setenumext` command.

```

5313 \cs_new_protected:Npn \__enumext_set_parse:n #1
5314 {
5315     \tl_set:Ne \__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
5316     \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
5317     { \tl_remove_all:Nn \__enumext_setkey_tmpb_tl {##1} }
5318     \tl_if_empty:NNTF \__enumext_setkey_tmpb_tl
5319     {
5320         \seq_put_right:Ne \__enumext_setkey_tmpa_seq
5321         { \tl_trim_spaces:n {#1} }
5322     }
5323     { \__enumext_set_error:nn {#1} { } }
5324 }
5325 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
5326 { \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 5327 \prop_const_from_keyval:Nn \c__enumext_meta_paths_prop
__enumext_add_meta_key:nnn 5328 {
__enumext_def_meta_key:nnn 5329   {enumext,1} = level-1,
__enumext_def_meta_key:Vnn 5330   {enumext,2} = level-2,
5331   {enumext,3} = level-3,
5332   {enumext,4} = level-4,
5333   {enumext*} = enumext*
5334 }
```

Now we create the user command taking care that unknown cannot be passed as an argument.

```
5335 \NewDocumentCommand \setenumextmeta { s O{enumext,1} m +m }
5336 {
5337   \str_if_eq:eeTF { \tl_trim_spaces:n {#3} } { unknown }
5338   { \msg_error:nn { enumext } { prohibited-unknown } }
5339   {
5340     \bool_if:nTF {#1}
5341     {
5342       \int_step_inline:nn { 4 }
5343       { \__enumext_add_meta_key:nnn { enumext, ##1 } {#3} {#4} }
5344       \__enumext_add_meta_key:nnn { enumext* } {#3} {#4}
5345     }
5346     { \__enumext_add_meta_key:nnn {#2} {#3} {#4} }
5347   }
5348 }
```

The internal functions `__enumext_add_meta_key:nnn` and `__enumext_def_meta_key:nnn` will check the *optional argument* and create the “meta-key”.

```
5349 \cs_new_protected:Npn \__enumext_add_meta_key:nnn #1
5350 {
5351   \tl_set:Nn \l__enumext_meta_path_tl {#1}
5352   \tl_replace_all:Nnn \l__enumext_meta_path_tl { ~ } {}
5353   \prop_get:NVNTF
5354   \c__enumext_meta_paths_prop \l__enumext_meta_path_tl \l__enumext_meta_path_tl
5355   { \__enumext_def_meta_key:Vnn \l__enumext_meta_path_tl }
5356   {
5357     \msg_error:nnn { enumext } { unknown-set } {#1}
5358     \use_none:nn
5359   }
5360 }
5361 \cs_new_protected:Npn \__enumext_def_meta_key:nnn #1#2#3
5362 {
5363   \bool_lazy_or:nnTF
5364   { \keys_if_exist_p:nn { enumext / #1 } {#2} }
5365   { \keys_if_exist_p:nn { enumext / enumext* } {#2} }
5366   { \msg_error:nnn { enumext } { already-defined } {#2} }
5367   {
5368     \keys_define:nn { enumext / #1 }
5369     {
5370       #2 .meta:n = {#3},
5371       #2 .value_forbidden:n = true
5372     }
5373   }
5374 }
5375 \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
```

We define a set of *keys* for command and we will save the default values of these in `\g__enumext_-foreach_default_key_tl` to avoid the use of group.

```
\__enumext_parse_foreach_keys:nn
__enumext_parse_foreach_keys:n 5376 \keys_define:nn { enumext / foreach }
\__enumext_foreach_keyans:nn 5377 {
__enumext_foreach_add_body:n 5378   before .tl_set:N = \l__enumext_foreach_before_tl,
5379   before .value_required:n = true,
5380   after .tl_set:N = \l__enumext_foreach_after_tl,
5381   after .value_required:n = true,
```

```

5382   start   .int_set:N = \l__enumext_foreach_start_int,
5383   start   .value_required:n = true,
5384   stop    .int_set:N = \l__enumext_foreach_stop_int,
5385   stop    .value_required:n = true,
5386   step    .int_set:N = \l__enumext_foreach_step_int,
5387   step    .value_required:n = true,
5388   wrapper .cs_set_protected:Np = \l__enumext_foreach_wrapper:n #1,
5389   wrapper .value_required:n = true,
5390   sep     .tl_set:N = \l__enumext_foreach_sep_tl,
5391   sep     .value_required:n = true,
5392   unknown .code:n = { \l__enumext_parse_foreach_keys:n {#1} }
5393 }
5394 \keys_precompile:nnN { enumext / foreach }
5395 {
5396   before={},after={},start=1,step=1,stop=0,wrapper=#1,sep={; }
5397 }
5398 \g__enumext_foreach_default_keys_tl

```

Functions for handling unknown $\langle keys \rangle$.

```

5399 \cs_new_protected:Npn \l__enumext_parse_foreach_keys:nn #1#2
5400 {
5401   \tl_if_blank:nTF {#2}
5402   {
5403     \msg_error:nnn { enumext } { for-key-unknown } {#1}
5404   }
5405   {
5406     \msg_error:nnnn { enumext } { for-key-value-unknown } {#1} {#2}
5407   }
5408 }
5409 \cs_new_protected:Npn \l__enumext_parse_foreach_keys:n #1
5410 {
5411   \exp_args:NV \l__enumext_parse_foreach_keys:nn \l_keys_key_str {#1}
5412 }

```

We create the command.

```

5413 \NewDocumentCommand \foreachkeyans { +0{ } m }
5414 {
5415   \l__enumext_foreach_keyans:nn {#1} {#2}
5416 }

```

Finally the internal functions `\l__enumext_foreach_keyans:nn` and `\l__enumext_foreach_add_body:n` will loop through the prop list and print the contents.

```

5417 \cs_new_protected:Npn \l__enumext_foreach_keyans:nn #1 #2
5418 {
5419   \tl_use:N \g__enumext_foreach_default_keys_tl
5420   \keys_set:nn { enumext / foreach } {#1}
5421   \tl_set:Nn \l__enumext_foreach_name_prop_tl {#2}
5422   \prop_if_exist:cF { g__enumext_#2_prop }
5423   {
5424     \msg_error:nnn { enumext } { undefined-storage-anskey } {#2}
5425   }
5426   \int_compare:nNnT { \l__enumext_foreach_stop_int } = { 0 }
5427   {
5428     \int_set:Nn \l__enumext_foreach_stop_int
5429     { \prop_count:c { g__enumext_#2_prop } }
5430   }
5431   \seq_clear:N \l__enumext_foreach_print_seq
5432   \int_step_function:nnnN
5433   { \l__enumext_foreach_start_int }
5434   { \l__enumext_foreach_step_int }
5435   { \l__enumext_foreach_stop_int }
5436   \l__enumext_foreach_add_body:n
5437   \seq_use:NV \l__enumext_foreach_print_seq \l__enumext_foreach_sep_tl
5438 }
5439 \cs_new_protected:Npn \l__enumext_foreach_add_body:n #1
5440 {
5441   \seq_put_right:Ne \l__enumext_foreach_print_seq
5442   {
5443     \exp_not:V \l__enumext_foreach_before_tl
5444     \l__enumext_foreach_wrapper:n
5445     {
5446       \prop_item:cn { g__enumext_ \l__enumext_foreach_name_prop_tl _prop } {#1}

```



```

5447     }
5448     \exp_not:V \l__enumext_foreach_after_tl
5449   }
5450 }

```

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

13.51 Messages

Message used by package-load for `multicol` and `hyperref` packages.

```

5451 \msg_new:nnn { enumext } { package-load }
5452 {
5453   The ~ '#1' ~ package ~ is ~ already ~ loaded.
5454 }
5455 \msg_new:nnn { enumext } { package-not-load }
5456 {
5457   The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
5458 }
5459 \msg_new:nnn { enumext } { package-load-foot }
5460 {
5461   The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
5462 }

```

Message used in the creation of counters by `enumext` package.

```

5463 \msg_new:nnn { enumext } { counters }
5464 {
5465   The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
5466   package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
5467 }

```

Message used by `align` and `mark-pos` keys.

```

5468 \msg_new:nnn { enumext } { unknown-choice }
5469 {
5470   The ~ value ~ '#3' ~ for ~ '#1' ~ key ~ is ~ invalid ~ use ~ ('#2').
5471 }

```

Message used by reserved `anskey*` environment by `enumext` package.

```

5472 \msg_new:nnnn { enumext } { anskey-env-error }
5473 {
5474   The ~ '#1' ~ environment ~ is ~ reserved ~ by ~ \\
5475   'enumext' ~ package, ~ It~ is~ already~ defined.
5476 }
5477 {
5478   The ~ anskey* ~ environment ~ is ~ defined ~ internally ~
5479   for ~ the ~ 'save-ans' ~ key.\\
5480 }

```

Message used in the creation of *prop list* by `enumext` package.

```

5481 \msg_new:nnn { enumext } { store-prop }
5482 {
5483   * ~ Package ~ enumext: ~ Creating ~
5484   \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
5485 }
5486 \msg_new:nnn { enumext } { store-seq }
5487 {
5488   * ~ Package ~ enumext: ~ Creating ~
5489   \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
5490 }
5491 \msg_new:nnn { enumext } { store-int }
5492 {
5493   * ~ Package ~ enumext: ~ Creating ~
5494   \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_context:.
5495 }
5496 \msg_new:nnn { enumext } { prop-seq-int-hook }
5497 {
5498   * ~ Package ~ enumext: ~ Elements ~ in ~
5499   \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
5500   * ~ Package ~ enumext: ~ Elements ~ in ~
5501   \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
5502   * ~ Package ~ enumext: ~ Value ~ off ~
5503   \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
5504 }
5505 \msg_new:nnn { enumext } { item-answer-hook }

```

```

5506 {
5507     * ~ Package ~ enumext: ~ Value ~ off ~
5508     \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\\
5509     * ~ Package ~ enumext: ~ Value ~ off ~
5510     \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\\
5511     * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
5512 }

```

Message used by [`key = val`] system and `\setenumext` command.

```

5513 \msg_new:nnn { enumext } { invalid-key }
5514 {
5515     The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
5516 }
5517 \msg_new:nnn { enumext } { unknown-key-family }
5518 {
5519     Unknown~key~family~`\l_keys_key_str'~for~enumext.
5520 }

```

Messages used in length calculation.

```

5521 \msg_new:nnn { enumext } { width-negative }
5522 {
5523     Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\
5524     The ~ key ~ '#1'~ accepts ~ values ~ >= ~ opt.
5525 }
5526 \msg_new:nnn { enumext } { width-zero }
5527 {
5528     Invalid ~ '#1=#2' ~ \msg_line_context:.\
5529     The ~ key ~ '#1'~ accepts ~ values ~ > ~ opt.
5530 }

```

Messages used by `show-length` key in `enumext`.

```

5531 \msg_new:nnn { enumext } { list-lengths }
5532 {
5533     **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
5534     \__enumext_show_length:nnn { dim } { labelsep } {#1}
5535     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
5536     \__enumext_show_length:nnn { dim } { itemindent } {#1}
5537     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
5538     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
5539     \__enumext_show_length:nnn { dim } { listparindent } {#1}
5540     \__enumext_show_length:nnn { skip } { topsep } {#1}
5541     \__enumext_show_length:nnn { skip } { parsep } {#1}
5542     \__enumext_show_length:nnn { skip } { partopsep } {#1}
5543     \__enumext_show_length:nnn { skip } { itemsep } {#1}
5544     *****
5545 }

```

Messages used by `show-length` key in `enumext*`, `keyans*` and `keyans`.

```

5546 \msg_new:nnn { enumext } { list-lengths-not-nested }
5547 {
5548     **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
5549     \__enumext_show_length:nnn { dim } { labelsep } {#1}
5550     \__enumext_show_length:nnn { dim } { labelwidth } {#1}
5551     \__enumext_show_length:nnn { dim } { itemindent } {#1}
5552     \__enumext_show_length:nnn { dim } { leftmargin } {#1}
5553     \__enumext_show_length:nnn { dim } { rightmargin } {#1}
5554     \__enumext_show_length:nnn { dim } { listparindent } {#1}
5555     \__enumext_show_length:nnn { skip } { topsep } {#1}
5556     \__enumext_show_length:nnn { skip } { parsep } {#1}
5557     \__enumext_show_length:nnn { skip } { partopsep } {#1}
5558     \__enumext_show_length:nnn { skip } { itemsep } {#1}
5559     *****
5560 }

```

Messages used by `ref` key.

```

5561 \msg_new:nnn { enumext } { key-ref-empty }
5562 {
5563     Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
5564 }

```

Messages used by `save-ans` key.

```

5565 \msg_new:nnn { enumext } { save-ans-empty }
5566 {
5567     Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.

```

```

5568 }
5569 \msg_new:nnn { enumext } { save-ans-log }
5570 {
5571   * ~ Package ~ enumext: ~ Start ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5572 }
5573 \msg_new:nnn { enumext } { save-ans-log-hook }
5574 {
5575   * ~ Package ~ enumext: ~ Stop ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5576 }
5577 \msg_new:nnn { enumext } { save-ans-hook }
5578 {
5579   Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
5580 }

```

Messages used by the internal system to check answer used by `check-ans` key.

```

5581 \msg_new:nnn { enumext } { need-save-ans }
5582 {
5583   Key ~ '#1' ~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2' ~ \msg_line_context:.
5584 }
5585 \msg_new:nnn { enumext } { items-same-answer }
5586 {
5587   *****\
5588   * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~
5589   for ~ \c_left_brace_str #2 \c_right_brace_str\
5590   * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~
5591   'OK', ~ all ~ items ~ with ~ answer.\
5592   *****
5593 }
5594 \msg_new:nnn { enumext } { item-greater-answer }
5595 {
5596   Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\
5597   started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\
5598   Items ~ > ~ Answers.
5599 }
5600 \msg_new:nnn { enumext } { item-less-answer }
5601 {
5602   Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\
5603   started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\
5604   Items ~ < ~ Answers.
5605 }

```

Messages used by the internal system to check for “starred” `\item*` and `\anspic*` commands.

```

5606 \msg_new:nnn { enumext } { missing-starred }
5607 {
5608   Missing ~ '\c_backslash_str #1*' ~ #2.
5609 }
5610 \msg_new:nnn { enumext } { many-starred }
5611 {
5612   Many ~ '\c_backslash_str #1*' ~ #2.
5613 }

```

Messages used by `\printkeyans*` command.

```

5614 \msg_new:nnn { enumext } { print-starred }
5615 {
5616   \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
5617   #2 ~ environment ~ \msg_line_context:.
5618 }

```

Message for the nesting depth of the environment `enumext`.

```

5619 \msg_new:nnn { enumext } { list-too-deep }
5620 {
5621   Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:~ \
5622   The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
5623 }

```

Messages used by `\anskey`, `anskey*` and `\anspic` commands.

```

5624 \msg_new:nnn { enumext } { anskey-unnumber-item }
5625 {
5626   Can't ~ store ~ with ~ a ~ unnumbered ~ \c_backslash_str item ~ \msg_line_context:.
5627 }
5628 \msg_new:nnn { enumext } { anskey-already-stored }
5629 {
5630   Content ~ already ~ stored ~ for ~ this ~ \c_backslash_str item ~ \msg_line_context:.

```

```

5631 }
5632 \msg_new:nnn { enumext } { anskey-empty-arg }
5633 {
5634   Can't ~ store ~ empty ~ content ~ \msg_line_context:.
5635 }
5636 \msg_new:nnn { enumext } { anskey-wrong-place }
5637 {
5638   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5639   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5640 }
5641 \msg_new:nnn { enumext } { anskey-nested }
5642 {
5643   The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
5644 }
5645 \msg_new:nnn { enumext } { anskey-math-mode }
5646 {
5647   #1 ~ can't ~ work ~ in ~ math ~ mode ~ \msg_line_context:.
5648 }
5649 \msg_new:nnn { enumext } { anskey-env-wrong }
5650 {
5651   The ~ environment ~ anskey* ~ cannot ~ use ~ in ~ '#1' ~ \msg_line_context:.
5652 }
5653 \msg_new:nnn { enumext } { anspic-wrong-place }
5654 {
5655   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5656   '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5657 }
5658 \msg_new:nnn { enumext } { command-wrong-place }
5659 {
5660   Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:~ \\
5661   '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
5662 }
5663 \msg_new:nnnn { enumext } { anskey-env-key-unknown }
5664 {
5665   The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5666   'anskey*' ~ and ~ is ~ being ~ ignored.
5667 }
5668 {
5669   The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5670   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5671 }
5672 \msg_new:nnnn { enumext } { anskey-env-key-value-unknown }
5673 {
5674   The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5675   'anskey*' ~ and ~ is ~ being ~ ignored.
5676 }
5677 {
5678   The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5679   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5680 }
5681 \msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
5682 { The ~ key ~ '#1'~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.}
5683 {
5684   The ~ command ~'\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5685   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5686 }
5687 \msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
5688 { The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.}
5689 {
5690   The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5691   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5692 }

```

Messages used by **keyans**, **keyans*** and **keyanspic** environment.

```

5693 \msg_new:nnn { enumext } { keyans-nested }
5694 {
5695   The ~ environment ~ 'keyans' ~ can't ~ be ~ nested ~ \msg_line_context:.
5696 }
5697 \msg_new:nnn { enumext } { keyans-wrong-level }
5698 {
5699   Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:~ \\
5700   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.

```

```

5701 }
5702 \msg_new:nnn { enumext } { wrong-place }
5703 {
5704   Wrong ~ place ~ for ~ '#1' ~ environment ~\msg_line_context::~ \
5705   '#1' ~ is ~ only ~ found ~ with ~ '#2' ~ in ~ 'enumext.
5706 }
5707 \msg_new:nnn { enumext } { keyanspic-nested }
5708 {
5709   The ~ environment ~ 'keyanspic' ~ can't ~ be ~ nested~ \msg_line_context::~.
5710 }
5711 \msg_new:nnn { enumext } { keyanspic-wrong-level }
5712 {
5713   Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context::~ \
5714   The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5715 }
5716 \msg_new:nnn { enumext } { keyanspic-item-cmd }
5717 {
5718   Can't ~ use ~ \c_backslash_str item ~ in ~ keyanspic ~ \msg_line_context:.
5719 }
5720 \msg_new:nnnn { enumext } { keyans-unknown-key }
5721 {
5722   The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5723   '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5724 }
5725 {
5726   The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5727   ~ have ~ a ~ key ~ called ~ '#1'.\
5728   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5729 }
5730 \msg_new:nnnn { enumext } { keyans-unknown-key-value }
5731 {
5732   The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5733   '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5734 }
5735 {
5736   The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5737   ~ have ~ a ~ key ~ called ~ '#1'.\
5738   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5739 }

```

Message used by unknown $\langle keys \rangle$ in `enumext*` environment.

```

5740 \msg_new:nnnn { enumext } { starred-unknown-key }
5741 {
5742   The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5743   '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5744 }
5745 {
5746   The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5747   ~ have ~ a ~ key ~ called ~ '#1'.\
5748   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5749 }
5750 \msg_new:nnnn { enumext } { starred-unknown-key-value }
5751 {
5752   The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5753   '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5754 }
5755 {
5756   The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5757   ~ have ~ a ~ key ~ called ~ '#1'.\
5758   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5759 }

```

Message used by unknown $\langle keys \rangle$ in `enumext` environment.

```

5760 \msg_new:nnnn { enumext } { standar-unknown-key }
5761 {
5762   The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_tl
5763   ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5764 }
5765 {
5766   The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5767   ~ have ~ a ~ key ~ called ~ '#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\
5768   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.

```

```

5769 }
5770 \msg_new:nnnn { enumext } { standar-unknown-key-value }
5771 {
5772   The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~ '\_enumext_envir_name_tl' \c_space_
5773   ~ on ~ level ~ \int_use:N \_enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5774 }
5775 {
5776   The ~ environment ~ '\_enumext_envir_name_tl' ~ does ~ not
5777   ~ have ~ a ~ key ~ called ~ '#1' ~ on ~ level ~ \int_use:N \_enumext_level_int.\
5778   Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5779 }

```

Message used by unknown *<keys>* in `\foreachkeyans`.

```

5780 \msg_new:nnnn { enumext } { for-key-unknown }
5781 { The~key~'#1'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored.}
5782 {
5783   The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\
5784   Check~that~you~have~spelled~the~key~name~correctly.
5785 }
5786 \msg_new:nnnn { enumext } { for-key-value-unknown }
5787 { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored. }
5788 {
5789   The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\
5790   Check~that~you~have~spelled~the~key~name~correctly.
5791 }

```

Messages used by `\getkeyans` command.

```

5792 \msg_new:nnn { enumext } { undefined-storage-anskey }
5793 {
5794   Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
5795 }

```

Messages used by `\miniright` command.

```

5796 \msg_new:nnn { enumext } { missing-miniright }
5797 {
5798   Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\
5799   The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
5800 }
5801 \msg_new:nnn { enumext } { wrong-miniright-place }
5802 {
5803   Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \
5804   Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
5805 }
5806 \msg_new:nnn { enumext } { wrong-miniright-use }
5807 {
5808   Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:~ \
5809   '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
5810 }
5811 \msg_new:nnn { enumext } { wrong-miniright-starred }
5812 {
5813   Can't ~ use ~ \c_backslash_str miniright ~ in ~ starred ~ environments ~ \msg_line_context:.
5814 }
5815 \msg_new:nnn { enumext } { many-miniright-used }
5816 {
5817   Can't ~ use ~ \c_backslash_str miniright ~ more ~ than ~ once ~ \msg_line_context:.
5818 }

```

Messages used by `\setenumextmeta` command.

```

5819 \msg_new:nnn { enumext } { unknown-set }
5820 {
5821   Argument ~ [#1] ~ is ~ unknown ~ by ~ \c_backslash_str setenumextmeta ~ \msg_line_context:.
5822 }
5823 \msg_new:nnn { enumext } { already-defined }
5824 {
5825   The ~ key ~ '#1' ~ is ~ already ~ defined ~ \msg_line_context:.
5826 }
5827 \msg_new:nnn { enumext } { prohibited-unknown }
5828 {
5829   The ~ name ~ 'unknown' ~ can't ~ be ~ chosen~ for ~ a ~ meta ~ key ~ \msg_line_context:.
5830 }

```

Messages used by `enumext*` and `keyans*` environments.

```

5831 \msg_new:nnn { enumext } { nested }

```

```

5832 {
5833   The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ \msg_line_con
5834 }
5835 \msg_new:nnn { enumext } { nested-horizontal }
5836 {
5837   The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ in ~ '#1' ~
5838 }
5839 \msg_new:nnn { enumext } { item-joined }
5840 {
5841   Items ~ joined ~ (#1) ~ > ~ #2 ~ columns ~\msg_line_context:.
5842 }
5843 \msg_new:nnn { enumext } { item-joined-columns }
5844 {
5845   Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.
5846 }

```

13.52 Finish package

Finish package implementation.

```

5847 \file_input_stop:
5848 </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>	227
<code>\+</code>	219
<code>\-</code>	219
<code>\\</code> 235, 2900, 4279, 4282, 5465, 5474, 5479, 5499, 5501, 5508, 5510, 5523, 5528, 5533, 5548, 5587, 5589, 5591, 5596, 5597, 5602, 5603, 5621, 5638, 5655, 5660, 5669, 5678, 5684, 5690, 5699, 5704, 5713, 5727, 5737, 5747, 5757, 5767, 5777, 5783, 5789, 5798, 5803, 5808	
A	
above	<u>1714</u>
above*	<u>1714</u>
<code>\addvspace</code> 1281, 1309, 1352, 1355, 1523, 1526, 1623, 1629, 1667, 1673, 1694, 1700, 3731, 3892, 3910, 4170, 4174, 4527, 4542, 4588, 4602	
after	<u>1111</u>
align	<u>664</u>
<code>\Alph</code>	40, 45, <u>46</u>
<code>\Alph</code>	606, 734, 778, 844, 5173
<code>\alph</code>	40, 45, <u>46</u>
<code>\alph</code>	607, 732, 5165
<code>\anskey</code>	13, 79, 81, <u>2718</u>
anskey*	13, <u>2828</u>
<code>\anspic</code>	16, 108, 111, <u>4184</u>
<code>\anspic*</code>	73
<code>\arabic</code>	32, 40
<code>\arabic</code>	605, 731, 777, 5157, 5161, 5177
B	
base-fix	<u>973</u>
<code>\baselineskip</code>	54
<code>\baselineskip</code>	989, 996
before	<u>1111</u>
before*	<u>1111</u>
below	<u>1714</u>
below*	<u>1714</u>
bool commands:	
<code>\bool_gset_false:N</code> 348, 349, 350, 3004, 3006, 4544, 4548, 4604	
<code>\bool_gset_true:N</code> 256, 266, 1214, 2207, 2213, 4513, 4545, 4577, 4605	
<code>\bool_if:NTF</code> . 399, 409, 426, 500, 507, 516, 523, 537, 550, 1736, 1750, 1763, 1774, 1785, 1796, 1807, 1818, 1867, 1884, 1889, 1897, 1924, 1962, 1967, 1974, 1978, 2000, 2005, 2013, 2020, 2051, 2059, 2152, 2350, 2360, 2439, 2463, 2470, 2494, 2592, 2614, 2654, 2668, 2672, 2722, 2741, 2765, 2819, 2830, 2919, 2956, 3020, 3055, 3070, 3145, 3156, 3160, 3179, 3192, 3234, 3268, 3304, 3319, 3340, 3477, 3492, 3509, 3572, 3582, 3615, 3620, 3686, 3712, 3762, 3820, 3875, 3900, 4104, 4168, 4186, 4205, 4250, 4277, 4506, 4522, 4528, 4571, 4585, 4589, 4678, 4688, 4776, 4782, 4789, 4805, 4898, 4908, 5019, 5023, 5049, 5056, 5084	
<code>\bool_if:nTF</code> 1674, 1701, 3290, 3458, 3530, 4226, 5196, 5340	
<code>\bool_if_p:N</code> 275, 290, 983, 984, 992, 993, 1646, 2031, 2032, 2040, 2041, 2165, 2191, 2204, 2205, 2210, 2211, 2527, 2537, 2549, 2564, 2565, 2599, 2640, 2641, 2942, 3132, 3133, 3170, 3171, 3659, 3661, 3672, 4233, 4234	
<code>\bool_lazy_all:nTF</code> 273, 288, 981, 2163, 2189, 2525, 2534, 2547, 2562, 3657, 3670	
<code>\bool_lazy_and:nnTF</code> 252, 262, 991, 1638, 1645, 2030, 2039, 2203, 2209, 2598, 2605, 2639, 2783, 2795, 2941, 2947, 3131	
<code>\bool_lazy_or:nnTF</code> . . 2092, 2099, 3169, 4232, 5363	
<code>\bool_new:N</code> 30, 31, 32, 33, 34, 35, 36, 37, 60, 69, 93, 98, 99, 104, 105, 108, 127, 134, 135, 142, 149, 150, 155, 157, 158, 175, 187, 189	
<code>\bool_not_p:n</code> 253, 263, 985, 1647, 2536, 2600, 2606, 2943, 2948, 3660, 3673	
<code>\bool_set_eq:NN</code> 3243, 3437, 4729, 4972	
<code>\bool_set_false:N</code> 406, 1003, 2137, 2138, 2170, 2175, 2179, 2183, 2196, 2883, 3634, 3779, 3828, 3915, 4057, 4101, 4647, 4673, 4726, 4914, 4969, 5209, 5210	
<code>\bool_set_true:N</code> . 280, 281, 295, 296, 391, 394, 657, 1018, 1720, 1725, 1987, 2109, 2110, 2382, 2390, 2884, 3237, 3239, 3271, 3273, 3433, 3445, 3595, 3633, 3666, 3679, 3752, 3825, 3852, 4054, 4495, 4560, 4646, 4733, 4740, 4741, 4785, 4912, 4976, 4983, 4984, 5204, 5205	
box commands:	
<code>\box_dp:N</code> . . 1569, 1570, 1573, 1580, 1593, 1601, 1607, 1615, 4115, 4120, 4170, 4261	
<code>\box_ht:N</code> . . 1352, 1355, 1366, 1367, 1378, 1380, 1395, 1398, 1406, 1407, 1418, 1420, 1435, 1438, 1445, 1446, 1457, 1459, 1474, 1477, 1523, 1526, 1534, 1535, 1543, 1544, 1556, 1558	
<code>\box_ht_plus_dp:N</code> 4110, 4178, 4214	
<code>\box_new:N</code> 66, 145, 146, 182, 188	
<code>\box_use_drop:N</code> 4539, 4600, 4841, 5096	
<code>\box_wd:N</code> 613	
C	
<code>\c</code>	227, 228, 880, 882, 894, 896
<code>\catcode</code>	2900
<code>\cB</code>	228
<code>\cE</code>	228
<code>\centering</code>	1676, 1703, 4305, 4532, 4593
check-ans	<u>2129</u>
Document class:	
article	47
clist commands:	
<code>\clist_const:Nn</code>	194
<code>\clist_map_function:nN</code>	4288
<code>\clist_map_inline:Nn</code> . 663, 928, 1110, 1125, 1206, 1730	
<code>\clist_map_inline:nn</code> . 45, 56, 74, 82, 95, 107, 137, 166, 193, 641, 694, 714, 1023, 1044, 1220, 1836, 2076, 2143, 2329, 2347, 2379, 2522, 3064, 3362, 3374, 3414, 3559, 3562, 3590, 3602, 3605, 3625, 5316	
<code>\columnbreak</code>	79
<code>\columnbreak</code>	2602
columns	<u>1190</u>
columns-sep	<u>1190</u>
<code>\columnsep</code>	101
<code>\columnsep</code>	3707, 3873
<code>\columnseprule</code>	101
<code>\columnseprule</code>	3710, 3874
Commands provide by enumext :	
<code>\anskey</code> 30, 69, 70, 75, 76, 78, 80, 81, 88, 90, 100, 101, 120,	

- 129, 130, 137
- \anspic* 30, 31, 73, 76, 88, 89, 110, 111, 129, 130
- \anspic 30, 76, 108, 111, 137
- \foreachkeyans 133, 140
- \getkeyans 76, 129, 140
- \item* 30, 31, 73, 76, 88, 89, 91, 92, 95, 121, 126, 127, 129, 130
- \item 91, 95, 114, 120–122, 125, 126
- \miniright 29, 52, 60, 61, 102, 140
- \printkeyans* 129
- \printkeyans 30, 76, 129, 130
- \setenumextmeta 132, 140
- \setenumext 30, 130–132, 136
- Counters defined by **enumext**:
- enumXiii 28, 40
- enumXii 28, 40
- enumXiv 28, 40
- enumXi 28, 40
- enumXviii 28, 40
- enumXvii 28, 40, 121
- enumXvi 28, 40
- enumXv 28, 40
- cs commands:
- \cs_generate_variant:Nn . 199, 200, 615, 631, 886, 902, 2431, 2436, 2512, 2836, 3549, 4290, 5375
- \cs_if_exist:NTF 585
- \cs_if_free:NTF 2787, 2799
- \cs_new:Nn 213
- \cs_new:Npn . 231, 1837, 1846, 1854, 2394, 2403, 2411, 5224, 5233, 5242
- \cs_new_eq:NN . 375, 376, 381, 382, 411, 412, 415, 416
- \cs_new_protected:Nn . 223, 237, 245, 271, 304, 334, 340, 346, 352, 358, 366, 386, 434, 438, 456, 468, 486, 498, 514, 530, 543, 564, 754, 815, 866, 979, 1126, 1130, 1134, 1138, 1142, 1146, 1150, 1154, 1158, 1162, 1166, 1170, 1174, 1178, 1182, 1186, 1221, 1233, 1266, 1283, 1294, 1311, 1337, 1358, 1483, 1509, 1529, 1562, 1584, 1619, 1625, 1731, 1745, 1759, 1770, 1781, 1792, 1803, 1814, 1895, 1998, 2011, 2028, 2049, 2077, 2082, 2107, 2148, 2158, 2201, 2216, 2223, 2232, 2237, 2242, 2247, 2256, 2261, 2266, 2437, 2461, 2468, 2492, 2499, 2513, 2739, 2758, 2774, 2837, 2873, 2904, 2939, 2981, 3002, 3010, 3053, 3068, 3096, 3129, 3165, 3177, 3190, 3276, 3286, 3297, 3313, 3329, 3454, 3470, 3486, 3500, 3626, 3655, 3684, 3691, 3721, 3738, 3760, 3782, 3818, 3842, 3859, 3884, 3898, 3919, 4076, 4272, 4286, 4291, 4315, 4325, 4356, 4485, 4504, 4550, 4569, 4633, 4660, 4667, 4676, 4686, 4711, 4852, 4896, 4927, 4933, 4954, 5011, 5107
- \cs_new_protected:Npn 201, 205, 209, 419, 583, 600, 610, 616, 735, 779, 849, 873, 887, 1658, 1687, 1863, 1882, 1952, 1985, 2087, 2271, 2348, 2358, 2380, 2388, 2423, 2432, 2588, 2651, 2666, 2704, 2708, 2828, 2859, 2863, 2894, 3030, 3106, 3150, 3230, 3249, 3375, 3379, 3393, 3397, 3415, 3419, 3429, 3441, 3518, 3552, 3593, 3637, 3838, 4085, 4092, 4099, 4203, 4222, 4246, 4387, 4436, 4650, 4717, 4724, 4738, 4746, 4751, 4761, 4920, 4960, 4967, 4981, 4989, 5006, 5129, 5142, 5190, 5313, 5325, 5349, 5361, 5399, 5409, 5417, 5439
- \cs_new_protected_nopar:Nn . . . 3982, 4026, 4034, 4042, 4696, 4704, 4835, 4939, 4947, 5090
- \cs_new_protected_nopar:Npn . . 3974, 3990, 4767, 4813, 5039, 5064
- \cs_set:Npn 2523, 2560, 5135
- \cs_set_eq:NN 4623, 4624, 4815, 4885, 4886, 5066
- \cs_set_protected:Nn 1049, 1065, 1078, 1090
- \cs_set_protected:Npn 41, 50, 67, 75, 90, 96, 130, 162, 173, 632, 642, 664, 699, 715, 761, 903, 929, 1005, 1028, 1102, 1111, 1190, 1207, 1714, 1825, 2068, 2129, 2288, 2330, 2366, 2515, 3057, 3351, 3367, 3407, 3550, 3591
- \cs_to_str:N 602, 625
- \cs_undefine:N 2776, 2777, 2778, 2779
- D**
- \d 219
- \DeclareDocumentEnvironment 568
- dim commands:
- \dim_abs:n 3523, 3528
- \dim_add:Nn 4119, 4350, 4381
- \dim_compare:nNnTF 1051, 1067, 1080, 1092, 1370, 1382, 1410, 1422, 1449, 1461, 1538, 1546, 1660, 1689, 3520, 3525, 3531, 3537, 3539, 3541, 3696, 3743, 3846, 3863, 4094, 4327, 4343, 4358, 4374, 4487, 4552
- \dim_compare:nTF 2624, 2969, 3785, 3922
- \dim_eval:n 989, 4176, 4257
- \dim_gset_eq:NN 4496, 4561
- \dim_gzero:N 3008, 4547, 4607
- \dim_new:N 63, 70, 71, 72, 92, 139, 147, 148, 181, 183, 184, 190
- \dim_set:Nn 613, 1019, 3266, 3523, 3528, 3530, 3533, 3534, 3538, 3540, 3543, 3544, 3546, 3699, 3746, 3784, 3848, 3865, 3921, 4108, 4212, 4293, 4329, 4336, 4360, 4367, 4422, 4471, 4489, 4554, 4763
- \dim_set_eq:NN 722, 768, 837, 841, 3181, 3182, 3194, 3195, 3261, 3561, 3604, 3707, 3873, 4429, 4432, 4433, 4478, 4481, 4482, 4756, 4827, 5078
- \dim_sub:Nn 3790, 3927, 4345, 4376
- \dim_use:N 1052, 1060, 1661, 1671, 2502, 2505, 2510, 3281, 3283, 3336, 3697, 3701, 3702, 3704, 3744, 3749, 3750, 3756, 3787, 3792
- \dim_zero:N 3596, 3710, 3874, 4121
- \dim_zero_new:N 582
- \c_zero_dim 1054, 1068, 1081, 1093, 1661, 1689, 2626, 2971, 3520, 3525, 3531, 3538, 3697, 3744, 3787, 3846, 3863, 3924, 4094, 4327, 4343, 4358, 4374, 4487, 4552
- \dimeval 2295
- E**
- \end 2465, 2496, 3728, 3889, 4158, 4307, 5198, 5208, 5216
- end internal commands:
- \end__enumext_mini_page . 1669, 1696, 3771, 3909, 4511, 4575, 4601
- \endgroup 2900
- \endlist 376
- \endminipage 382
- enumext 5, 3796
- enumext internal commands:
- \l__enumext__ref_the_count_tl 43
- \l__enumext__resume_name_tl 65
- __enumext_add_meta_key:nnn . . . 133, 5327, 5343, 5344, 5346, 5349
- __enumext_add_pre_parsep: . 53, 1231, 1233, 1233
- __enumext_after_args_exec: 50, 1126, 1138, 3809
- __enumext_after_args_exec_v: 1142, 1154, 3942
- __enumext_after_args_exec_viii: . . 1158, 1182
- __enumext_after_args_exec_viiii: 1186
- __enumext_after_env:nn 85–87, 104, 116, 124, 205, 205, 556, 560, 2914, 3814, 4520, 4583, 4868
- __enumext_after_hyperref: . . . 36, 384, 384, 386
- \l__enumext_after_list_args_v_tl 1156

```

\l__enumext_after_list_args_vii_tl 1184, 4833
\l__enumext_after_list_args_viii_tl .. 1188,
  5088
\__enumext_after_list_vii: 116, 119, 4631, 4667,
  4667
\__enumext_after_list_viii: ... 125, 4894, 4933,
  4933
\__enumext_after_stop_list: 50, 103, 1126, 1134,
  3776
\__enumext_after_stop_list_v: 1142, 1150, 3916
\l__enumext_after_stop_list_v_tl ..... 1152
\__enumext_after_stop_list_vii: .. 119, 1158,
  1174, 4670
\l__enumext_after_stop_list_vii_tl ... 1176
\__enumext_after_stop_list_viii: . 1178, 4936
\l__enumext_after_stop_list_viii_tl ... 1180
\l__enumext_align_label_pos_v_str .... 3506
\l__enumext_align_label_pos_X_str ..... 75
\l__enumext_align_label_vii_str ..... 4802
\l__enumext_align_label_viii_str ..... 5053
\l__enumext_align_label_X_str ..... 173
\c__enumext_all_envs_clist . 194, 663, 928, 1110,
  1125, 1206, 1730
\c__enumext_all_families_seq .. 132, 5281, 5307
\l__enumext_anskey_env_bool 33, 84, 30, 281, 296,
  2830
\__enumext_anskey_env_clean_vars: . 87, 2935,
  2939, 3002
\__enumext_anskey_env_define_keys: 84, 2828,
  2837, 2908
\__enumext_anskey_env_exec: 85, 2833, 2904, 2904
\__enumext_anskey_env_make:n 69, 84, 2112, 2828,
  2828, 2836
\__enumext_anskey_env_reset_keys: 85, 86, 2828,
  2873, 2936
\__enumext_anskey_env_save_keys: .. 86, 2916,
  2939, 2939
\__enumext_anskey_env_store: .. 86, 2932, 2939,
  2981
\__enumext_anskey_env_unknown:n 84, 2828, 2856,
  2859
\__enumext_anskey_env_unknown:nn . 2828, 2861,
  2863
\l__enumext_anskey_level_int .. 24, 2760, 2761
\__enumext_anskey_safe_inner: . 82, 2733, 2739,
  2758
\__enumext_anskey_safe_inner:n ..... 82
\__enumext_anskey_safe_outer: . 81, 2720, 2739,
  2739
\__enumext_anskey_show_wrap_arg:n . 80, 2651,
  2651, 2670, 2685
\__enumext_anskey_show_wrap_left:n 80, 2596,
  2666, 2666
\__enumext_anskey_unknown:n 81, 2688, 2702, 2704
\__enumext_anskey_unknown:nn . 2688, 2706, 2708
\__enumext_anskey_wrapper:n ..... 2292, 2664
\l__enumext_anspic_above_int . 138, 4294, 4295,
  4297
\__enumext_anspic_args:nnn 111, 112, 4184, 4200,
  4272
\l__enumext_anspic_args_seq 111, 112, 138, 4198,
  4306, 4319
\l__enumext_anspic_below_int . 138, 4294, 4295,
  4298
\l__enumext_anspic_body_box ... 138, 4211, 4214
\__enumext_anspic_body_dim:n .. 111, 4184, 4203,
  4249
\l__enumext_anspic_body_htdp_dim .. 111, 138,
  4212, 4260
__enumext_anspic_exec: ..... 4184
\__enumext_anspic_exec: ... 110, 113, 4153, 4315
\__enumext_anspic_label:nn 111, 4184, 4222, 4252,
  4267
\l__enumext_anspic_label_above_bool ... 138,
  4054, 4057, 4104, 4168, 4205, 4250, 4277
\l__enumext_anspic_label_box .. 138, 4107, 4110
\l__enumext_anspic_label_htdp_dim . 109, 138,
  4108, 4114, 4178, 4259
\__enumext_anspic_label_pos:nnn .. 112, 4184,
  4246, 4275
\l__enumext_anspic_label_sep_skip 4064, 4116,
  4179, 4262, 4279
\l__enumext_anspic_layout_style_tl 4066, 4317,
  4322
\l__enumext_anspic_mini_pos_str .. 138, 4055,
  4058, 4304
\l__enumext_anspic_mini_width_dim 138, 4224,
  4293, 4304
\__enumext_anspic_print:n 112, 4184, 4286, 4290,
  4319, 4322
\__enumext_anspic_row:n .. 112, 4184, 4288, 4291
\__enumext_anspic_start_list_tag: 3998, 4026,
  4274
\__enumext_anspic_stop_list_tag: . 3998, 4042,
  4284
\__enumext_anspic_stop_start_list_tag: 3998,
  4034, 4276
\__enumext_at_begin_document:n .. 35, 201, 201,
  373, 379
\l__enumext_base_line_fix_bool 47, 48, 130, 975,
  984, 1003, 5204, 5209
\__enumext_before_args_exec: 50, 102, 119, 1126,
  1126, 3741
\__enumext_before_args_exec_v: 1142, 1142, 3845
\__enumext_before_args_exec_vii: . 1158, 1158,
  4664
\__enumext_before_args_exec_viii: 1162, 4930
\__enumext_before_env:nn 84, 205, 209, 2781, 2793,
  2805, 2906
\__enumext_before_keys_exec: .. 50, 1126, 1130,
  3806
\__enumext_before_keys_exec_v: 1142, 1146, 3939
\__enumext_before_keys_exec_vii ..... 1158
\__enumext_before_keys_exec_vii: . 1166, 4618
\__enumext_before_keys_exec_viii: 1170, 4880
\__enumext_before_list: .. 102, 3738, 3738, 3800
\__enumext_before_list_v: ... 3842, 3842, 3934
\__enumext_before_list_vii: ... 119, 4613, 4660,
  4660
\__enumext_before_list_viii: .. 125, 4876, 4927,
  4927
\l__enumext_before_no_starred_key_v_tl 1148
\l__enumext_before_no_starred_key_vii_-
  tl ..... 1168
\l__enumext_before_no_starred_key_viii_-
  tl ..... 1172
\l__enumext_before_starred_key_v_tl ... 1144
\l__enumext_before_starred_key_vii_tl . 1160
\l__enumext_before_starred_key_viii_tl 1164
\__enumext_calc_hspace:NNNNNNN 97, 3518, 3518,

```

3549, 3554, 3597
 __enumext_check_ans_active: 70, 102, 119, 2148,
 2148, 3742, 4663
 \g__enumext_check_ans_item_tl 89
 \g__enumext_check_ans_key_bool 71, 72, 149, 348,
 2207, 2213, 3020
 \l__enumext_check_ans_key_bool 71, 2133, 2138,
 2204, 2210
 __enumext_check_ans_key_hook: . . 71, 103, 119,
 2201, 2201, 3777, 4671
 __enumext_check_ans_level: . 70, 71, 2148, 2154,
 2158
 __enumext_check_ans_log: 72, 87, 2247, 2247, 3024
 __enumext_check_ans_log_msg_greater: 2247,
 2253, 2266
 __enumext_check_ans_log_msg_less: 2247, 2251,
 2256
 __enumext_check_ans_log_msg_same_ok: 2247,
 2252, 2261
 __enumext_check_ans_msg_greater: 2223, 2229,
 2242
 __enumext_check_ans_msg_less: 2223, 2227, 2232
 __enumext_check_ans_msg_same_ok: 2223, 2228,
 2237
 __enumext_check_ans_show: . . 72, 87, 2223, 2223,
 3022
 \l__enumext_check_answers_bool 69, 70, 81, 91, 92,
 149, 2110, 2137, 2152, 2439, 2463, 2470, 2494, 2722,
 2919, 3145, 3234, 3268, 4782
 __enumext_check_starred_cmd:n 34, 73, 89, 124,
 2271, 2271, 3945, 4166, 4893
 \g__enumext_check_starred_cmd_int . . 96, 149,
 2274, 2280, 2285, 3452, 4231, 5018
 \l__enumext_check_start_line_env_tl . 34, 149,
 311, 319, 327, 2277, 2283, 2286
 \l__enumext_columns_sep_v_dim 3863, 3865, 3873
 \l__enumext_columns_sep_vii_dim . . 4327, 4329,
 4338, 4350, 4426, 4849
 \l__enumext_columns_sep_viii_dim . 4358, 4360,
 4369, 4381, 4475, 5104
 \l__enumext_columns_v_int 1503, 1521, 1692, 3861,
 3869, 3881, 3886
 \l__enumext_columns_vii_int . . 4332, 4335, 4339,
 4348, 4390, 4394, 4397, 4403, 4409, 4413, 4843, 4857
 \l__enumext_columns_viii_int . 4363, 4366, 4370,
 4379, 4439, 4443, 4446, 4452, 4458, 4462, 5098, 5113
 \l__enumext_counter_i_tl 41, 592
 \l__enumext_counter_ii_tl 41, 593
 \l__enumext_counter_iii_tl 41, 594
 \l__enumext_counter_iv_tl 41, 595
 \c__enumext_counter_style_tl 32, 46, 225
 \g__enumext_counter_styles_tl . 28, 40, 63, 603,
 621
 \l__enumext_counter_v_tl 41, 596, 857
 \l__enumext_counter_vi_tl 41, 597
 \l__enumext_counter_vii_tl 41, 598, 789
 \l__enumext_counter_viii_tl 41, 599, 805
 \l__enumext_current_widest_dim 28, 63, 627, 723,
 769, 838, 842
 __enumext_def_meta_key:nnn . . . 133, 5327, 5355,
 5361, 5375
 __enumext_default_item:n . . . 3230, 3230, 3294
 __enumext_define_counters:Nn 28, 583, 583, 592,
 593, 594, 595, 596, 597, 598, 599
 __enumext_endminipage: . 36, 373, 382, 577, 4541,
 4837, 5092
 \g__enumext_envir_name_tl 33, 30, 282, 297, 356,
 2080, 2085, 2095, 2235, 2240, 2245, 2259, 2264, 2269
 \l__enumext_envir_name_tl . 33, 34, 30, 251, 261,
 310, 318, 326, 5723, 5726, 5733, 5736, 5743, 5746,
 5753, 5756, 5762, 5766, 5772, 5776, 5833, 5837
 __enumext_execute_after_env: 34, 35, 68, 72, 83,
 87, 3010, 3010, 3816, 4870
 __enumext_fake_item_indent: . 1049, 1049, 3581
 \l__enumext_fake_item_indent_v_dim 1068, 1073
 \l__enumext_fake_item_indent_v_tl 1070, 3434,
 3438, 3446
 __enumext_fake_item_indent_vii: . 1049, 1078,
 3614
 \l__enumext_fake_item_indent_vii_dim . 1081,
 1085
 \l__enumext_fake_item_indent_vii_tl . . 1083,
 4832
 __enumext_fake_item_indent_viii: 1049, 1090,
 3619
 \l__enumext_fake_item_indent_viii_dim 1093,
 1097
 \l__enumext_fake_item_indent_viii_tl . 1095,
 5083
 \l__enumext_fake_item_indent_X_tl 96
 __enumext_fake_make_label_vii:n . 121, 4767,
 4767, 4824
 __enumext_fake_make_label_viii:n 5039, 5039,
 5075
 __enumext_filter_first_level:n . . 131, 5224,
 5224, 5258, 5269
 __enumext_filter_first_level_key:n 131, 5224,
 5229, 5233
 __enumext_filter_first_level_pair:nn . 131,
5224, 5230, 5242
 __enumext_filter_save_key:n . . 75, 2355, 2363,
 2386, 2392, 2394, 2394, 5155, 5159, 5163, 5167, 5171,
 5175
 __enumext_filter_save_key_key:n . . 75, 2394,
 2399, 2403
 __enumext_filter_save_key_pair:nn 76, 2394,
 2400, 2411
 __enumext_filter_series:n 64, 1837, 1837, 1875,
 1887, 1892
 __enumext_filter_series_key:n 64, 1837, 1842,
 1846
 __enumext_filter_series_pair:nn . . 64, 1837,
 1843, 1854
 __enumext_first_item_tmp_vii: 118, 120, 4623,
4696, 4696
 __enumext_first_item_tmp_viii: . . 125, 4885,
4939, 4939
 \g__enumext_footnote_standar_arg_seq . . 167,
 451, 462, 465
 \g__enumext_footnote_standar_int 167, 445, 448,
 450, 453
 \g__enumext_footnote_standar_int_seq . . 167,
 453, 458, 461, 466
 \g__enumext_footnote_starred_arg_seq . . 167,
 481, 492, 495
 \g__enumext_footnote_starred_int 167, 475, 478,
 480, 483
 \g__enumext_footnote_starred_int_seq . . 167,
 483, 488, 491, 496
 __enumext_footnotes_key_bool 36

`\l__enumext_footnotes_key_bool` 31, 36, [157](#), 394, 399, 406, 507, 523, 537, 550
`__enumext_footnotetext:nn` .. [434](#), 434, 463, 493
`__enumext_foreach_add_body:n` 134, [5376](#), 5436, 5439
`\l__enumext_foreach_after_tl` 5380, 5448
`\l__enumext_foreach_before_tl` 5378, 5443
`\g__enumext_foreach_default_keys_tl` 133, [122](#), 5398, 5419
`__enumext_foreach_keyans:nn` . 134, [5376](#), 5415, 5417
`\l__enumext_foreach_name_prop_tl` . [122](#), 5421, 5446
`\l__enumext_foreach_print_seq` [122](#), 5431, 5437, 5441
`\l__enumext_foreach_sep_tl` 5390, 5437
`\l__enumext_foreach_start_int` 5382, 5433
`\l__enumext_foreach_step_int` 5386, 5434
`\l__enumext_foreach_stop_int` . 5384, 5426, 5428, 5435
`__enumext_foreach_wrapper:n` 5388, 5444
`__enumext_getkeyans:nn` .. 129, [5124](#), 5138, 5142
`__enumext_getkeyans_aux:n` 129, [5124](#), 5126, 5129
`\l__enumext_hyperref_bool` 31, 36, [157](#), 391, 409, 426, 2641, 3133, 4776
`__enumext_hypertarget:nn` 36, [384](#), 411, 415, 431
`__enumext_if_is_int:n` 217
`__enumext_if_is_int:nTF` [217](#), 875, 889
`__enumext_internal_mini_page:` 39, 100, 118, [564](#), 564, 3629, 4636
`__enumext_is_not_nested:` . 28, 33, 100, 118, [245](#), 245, 3628, 4635
`__enumext_is_on_first_level:` . 28, 33, 100, 119, [245](#), 271, 3635, 4648
`\g__enumext_item_anskey_int` 81, 90, [149](#), 343, 370, 371, 2220, 2590, 3147
`__enumext_item_answer_diff:` 72, 87, [2216](#), 2216, 3017
`\g__enumext_item_answer_diff_int` 72, [149](#), 344, 2218, 2225, 2249
`\l__enumext_item_column_pos_vii_int` 120, 4397, 4403, 4409, 4413, 4420, 4707, 4843, 4846
`\l__enumext_item_column_pos_viii_int` .. 125, 4446, 4452, 4458, 4462, 4469, 4950, 5098, 5101
`\l__enumext_item_column_pos_X_int` [173](#)
`\g__enumext_item_count_all_vii_int` 120, 4421, 4708, 4857, 4865
`\g__enumext_item_count_all_viii_int` 125, 4470, 4951, 5112, 5121
`\g__enumext_item_count_all_X_int` [173](#)
`\g__enumext_item_number_bool` [149](#)
`\l__enumext_item_number_bool` 71, 155, 2170, 2175, 2179, 2183, 2196, 2765, 2819, 3237, 3271, 4785
`\g__enumext_item_number_int` .. 71, [149](#), 342, 369, 371, 2169, 2174, 2178, 2182, 2195, 2220, 3236, 3270, 4784
`__enumext_item_peek_args_vii:` 120, [4704](#), 4709, 4711
`__enumext_item_peek_args_viii:` 125, 126, [4947](#), 4952, 4954
`__enumext_item_star_exec:` 92, [3249](#), 3276, 3321, 3342
`\l__enumext_item_starred_vii_bool` 4726, 4740, 4789
`\l__enumext_item_starred_viii_bool` 4969, 4983, 5049, 5084
`\l__enumext_item_starred_X_bool` [173](#)
`__enumext_item_std:w` 35, 91, 92, 96, [373](#), 377, 3240, 3246, 3274, 3434, 3438, 3446
`\g__enumext_item_symbol_aux_tl` . 92, [126](#), 3254, 3257, 3282, 3326, 3346
`\g__enumext_item_symbol_aux_vii_tl` 4748, 4791, 4794, 4798, 4800
`\g__enumext_item_symbol_aux_X_tl` [173](#)
`\l__enumext_item_symbol_sep_vii_dim` .. 4756, 4763, 4797, 4799
`\l__enumext_item_symbol_vii_tl` 4794
`\l__enumext_item_text_vii_box` 4816, 4841
`\l__enumext_item_text_viii_box` . . . 5067, 5096
`\l__enumext_item_text_X_box` [173](#)
`\l__enumext_item_width_vii_dim` . . 4336, 4345, 4424, 4432, 4433
`\l__enumext_item_width_viii_dim` .. 4367, 4376, 4473, 4481, 4482
`\l__enumext_item_width_X_dim` [173](#)
`\l__enumext_itemindent_X_dim` [67](#)
`\l__enumext_itemsep_i_skip` . . . 1364, 1371, 1374, 1376, 1383, 1387, 1390, 1392, 1532, 1539, 1541, 1542, 1547, 1551, 1553, 1554
`\l__enumext_itemsep_ii_skip` .. 1404, 1411, 1414, 1416, 1423, 1427, 1430, 1432
`\l__enumext_itemsep_iii_skip` . 1443, 1450, 1453, 1455, 1462, 1466, 1469, 1471
`\l__enumext_itemsep_vii_skip` 4863
`\l__enumext_itemsep_viii_skip` 5119
`\l__enumext_joined_item_aux_vii_int` .. 4418, 4419, 4420, 4421, 4427
`\l__enumext_joined_item_aux_viii_int` . 4467, 4468, 4469, 4470, 4476
`\l__enumext_joined_item_aux_X_int` [173](#)
`__enumext_joined_item_vii:w` .. 120, [4704](#), 4714, 4715, 4717
`\l__enumext_joined_item_vii_int` .. 4389, 4390, 4393, 4395, 4401, 4406, 4411, 4416, 4418, 4424
`__enumext_joined_item_viii:w` . 126, [4947](#), 4957, 4958, 4960
`\l__enumext_joined_item_viii_int` . 4438, 4439, 4442, 4444, 4450, 4455, 4460, 4465, 4467, 4473
`\l__enumext_joined_item_X_int` [173](#)
`\l__enumext_joined_width_vii_dim` . 4422, 4429, 4432, 4818, 4826
`\l__enumext_joined_width_viii_dim` 4471, 4478, 4481, 5069, 5077
`\l__enumext_joined_width_X_dim` [173](#)
`__enumext_keyans_addto_prop:n` 88, [3030](#), 3030, 3449, 4228
`__enumext_keyans_addto_seq:n` . 89, [3106](#), 3106, 3451, 4230
`__enumext_keyans_addto_seq_link:` [3106](#), 3127, 3129, 5017
`__enumext_keyans_default_item:n` .. 95, [3429](#), 3429, 3466
`\l__enumext_keyans_env_bool` [30](#), 3660, 3673, 3825, 3915
`__enumext_keyans_fake_item_indent:` .. [1049](#), 1065, 3571
`\l__enumext_keyans_level_h_int` .. 124, [24](#), 798, 824, 2749, 2811, 3084, 4642, 4902, 4903


```

\l__enumext_keyans_level_int .. 24, 1652, 2745,
    2807, 3079, 3824, 3829, 4194
\__enumext_keyans_make_label: . 96, 3470, 3470,
    3569
\__enumext_keyans_make_label_box: 3470, 3474,
    3479, 3500
\__enumext_keyans_make_label_std: 3470, 3482,
    3486
\__enumext_keyans_mini_right_cmd:n 61, 1654,
    1687, 1687
\__enumext_keyans_mini_set_vskip: ..... 57
\__enumext_keyans_minipage_add_space: 1483,
    1509, 3854
\__enumext_keyans_minipage_set_skip: . 1483,
    1483, 1511
\__enumext_keyans_multi_addvspace: 1283, 1294,
    3878
\__enumext_keyans_multi_set_vskip: 54, 1283,
    1283, 1296
\__enumext_keyans_multicols_start: 3842, 3857,
    3859
\__enumext_keyans_multicols_stop: 1691, 3842,
    3884, 3913
\__enumext_keyans_name_and_start: 28, 34, 124,
    304, 304, 3826, 4083, 4907
\__enumext_keyans_parse_keys:n 3838, 3838, 3933
\__enumext_keyans_pic_arg_two: 109, 4076, 4099,
    4129
\l__enumext_keyans_pic_level_int .. 24, 1633,
    2753, 2815, 3033, 3074, 3109, 3197, 4078, 4079
\__enumext_keyans_pic_parse_keys:n 4076, 4085,
    4128
\g__enumext_keyans_pic_parsep_skip ... 138
\__enumext_keyans_pic_safe_exec: . 108, 4076,
    4076, 4127
\__enumext_keyans_pic_skip_abs:N . 109, 4076,
    4092, 4103
\__enumext_keyans_pre_itemsep_skip: .. 1483,
    1502, 1529
\__enumext_keyans_redefine_item: .. 96, 3454,
    3454, 3568
\__enumext_keyans_ref: ..... 45, 849, 866, 3570
\__enumext_keyans_ref:n ..... 45, 846, 849, 849
\__enumext_keyans_safe_exec: . 3818, 3818, 3932
\__enumext_keyans_set_item_width: 105, 3919,
    3919, 3941
\__enumext_keyans_show_ans: .. 3150, 3158, 3177
\__enumext_keyans_show_item_opt: .. 96, 3150,
    3165, 3447, 4243, 5086
\__enumext_keyans_show_left:n . 96, 3150, 3150,
    3444, 4237
\__enumext_keyans_show_pos: .. 3150, 3162, 3190
\__enumext_keyans_starred_item:n .. 96, 3441,
    3441, 3462
\__enumext_keyans_store_ref: .. 88, 3053, 3053,
    3450, 4229, 5015
\__enumext_keyans_store_ref_aux_i: 88, 3053,
    3065, 3068
\__enumext_keyans_store_ref_aux_ii: 89, 3053,
    3094, 3096
\__enumext_keyans_unknown_keys:n . 3367, 3371,
    3375, 4074
\__enumext_keyans_unknown_keys:nn 3367, 3377,
    3379
\__enumext_keyans_wrapper_opt:n .. 2298, 3173
\l__enumext_label_copy_i_tl .. 2556, 3072, 3077,
    3082, 3087
\l__enumext_label_copy_v_tl ..... 3082
\l__enumext_label_copy_vi_tl ..... 3077
\l__enumext_label_copy_vii_tl 2532, 2543, 2572,
    3072
\l__enumext_label_copy_viii_tl ..... 3087
\l__enumext_label_copy_X_tl ..... 159
\l__enumext_label_fill_left_v_tl ..... 3490
\l__enumext_label_fill_left_X_tl ..... 96
\l__enumext_label_fill_right_v_tl .... 3497
\l__enumext_label_fill_right_X_tl ..... 96
\l__enumext_label_font_style_v_tl 3491, 3508,
    4241
\l__enumext_label_font_style_vii_tl ... 4804
\l__enumext_label_font_style_viii_tl .. 5055
\l__enumext_label_i_tl ..... 715
\l__enumext_label_ii_tl ..... 715
\l__enumext_label_iii_tl ..... 715
\l__enumext_label_iv_tl ..... 715
\__enumext_label_style:Nnn 28, 40, 616, 616, 631,
    720, 766, 835, 839
\l__enumext_label_v_tl 89, 832, 3038, 3114, 3184,
    3224, 3443, 3448, 3936, 4107, 4236, 4238
\l__enumext_label_vi_tl 89, 832, 3035, 3111, 4236,
    4238, 4242
\l__enumext_label_vii_tl . 761, 4735, 4758, 4765
\l__enumext_label_viii_tl 761, 4978, 5009, 5013
\l__enumext_label_width_by_box .. 63, 612, 613
\__enumext_label_width_by_box:Nn 40, 610, 610,
    615, 627, 899
\l__enumext_labelsep_i_dim ... 3182, 3187, 3195,
    3227, 5021, 5036
\l__enumext_labelsep_v_dim ..... 3868
\l__enumext_labelsep_vii_dim . 2657, 3182, 3195,
    4331, 4341, 4425, 4700, 4756, 4811, 4820
\l__enumext_labelsep_viii_dim 4362, 4372, 4474,
    4943, 5062, 5071
\l__enumext_labelwidth_i_dim . 3181, 3187, 3194,
    3227, 5021, 5036
\l__enumext_labelwidth_v_dim ..... 3506, 3868
\l__enumext_labelwidth_vii_dim ... 2657, 3181,
    3194, 4331, 4340, 4425, 4700, 4802, 4819
\l__enumext_labelwidth_viii_dim .. 4362, 4371,
    4474, 4943, 5053, 5070
\l__enumext_leftmargin_tmp_v_bool . 109, 4101
\l__enumext_leftmargin_tmp_X_bool ..... 67
\l__enumext_leftmargin_tmp_X_dim ..... 67
\l__enumext_leftmargin_X_dim ..... 67
\__enumext_level: 213, 213, 744, 747, 748, 756, 758,
    1052, 1056, 1060, 1128, 1132, 1136, 1140, 1223, 1225,
    1227, 1229, 1271, 1273, 1275, 1277, 1281, 1315, 1321,
    1326, 1328, 1331, 1334, 1347, 1350, 1661, 1665, 1671,
    1734, 1736, 1738, 1741, 1748, 1750, 1752, 1755, 2350,
    2352, 2354, 2382, 2383, 2385, 2441, 2449, 2453, 2457,
    2661, 2662, 3239, 3240, 3244, 3245, 3246, 3254, 3262,
    3263, 3266, 3273, 3274, 3278, 3281, 3283, 3317, 3318,
    3319, 3322, 3325, 3336, 3337, 3339, 3340, 3343, 3666,
    3679, 3686, 3694, 3697, 3699, 3701, 3702, 3703, 3704,
    3707, 3712, 3718, 3724, 3731, 3744, 3746, 3749, 3750,
    3752, 3756, 3762, 3787, 3792, 3803, 3805
\l__enumext_level_h_int 119, 24, 254, 277, 291, 782,
    817, 1640, 2166, 2186, 2551, 2785, 2797, 3674, 4637,

```

4638
 \l__enumext_level_int . 100, 24, 215, 264, 276, 292,
 566, 1235, 1360, 1639, 2160, 2192, 2528, 2538, 2544,
 2550, 2557, 2566, 2571, 2784, 2796, 3012, 3585, 3630,
 3631, 3642, 3650, 3664, 3677, 3708, 3833, 4190, 4680,
 4690, 4915, 5763, 5767, 5773, 5777
 __enumext_list_arg_two_i: 3550
 __enumext_list_arg_two_ii: 3550
 __enumext_list_arg_two_iii: 3550
 __enumext_list_arg_two_iv: 3550
 __enumext_list_arg_two_v: . 96, 3550, 3938, 4102
 __enumext_list_arg_two_vii: 3591, 4617
 __enumext_list_arg_two_viii: 3591, 4879
 \l__enumext_listoffset_v_dim . 3870, 3924, 3927
 \l__enumext_listparindent_vii_dim 4827, 4831
 \l__enumext_listparindent_viii_dim 5078, 5082
 __enumext_log_answer_vars: . 35, 358, 366, 3019
 __enumext_log_global_vars: . 35, 358, 358, 3018
 __enumext_make_label: 93, 3297, 3297, 3579
 __enumext_make_label_box: . . . 3297, 3301, 3306,
 3329
 __enumext_make_label_std: . . . 3297, 3309, 3313
 \l__enumext_mark_answer_sym_tl 77, 2304, 2507,
 2674, 3199, 3212, 5025
 \l__enumext_mark_position_str 126, 2308, 2309,
 2335, 2336, 2505
 \l__enumext_mark_ref_sym_tl . 2321, 2646, 3141
 \l__enumext_meta_path_tl . 122, 5351, 5352, 5354,
 5355
 \c__enumext_meta_paths_prop 133, 5327
 __enumext_mini_addvspace_vii: 59, 1619, 1619,
 4499
 __enumext_mini_addvspace_viii: 59, 1619, 1625,
 4564
 __enumext_mini_env* 564
 __enumext_mini_page 1671, 1698, 3756, 3855, 4501,
 4566, 4587
 __enumext_mini_right_cmd:n . 60, 61, 1656, 1658,
 1658
 __enumext_mini_set_vskip_vii: 58, 1562, 1562,
 1621
 __enumext_mini_set_vskip_viii: 58, 1562, 1584,
 1627
 __enumext_minipage:w 36, 373, 381, 571, 4524, 4826,
 5077
 \l__enumext_minipage_active_v_bool 3852, 3875,
 3900
 \g__enumext_minipage_active_vii_bool . . 116,
 4513, 4522, 4544
 \l__enumext_minipage_active_vii_bool . 4495,
 4506
 \g__enumext_minipage_active_viii_bool 4577,
 4585, 4604
 \l__enumext_minipage_active_viii_bool 4560,
 4571
 \g__enumext_minipage_active_X_bool . . . 173
 \l__enumext_minipage_active_X_bool 83
 __enumext_minipage_add_space: . 55, 102, 1311,
 1337, 3754
 \g__enumext_minipage_after_skip 83, 1566, 1578,
 4542, 4602
 \l__enumext_minipage_after_skip . . 54, 102, 83,
 1324, 1364, 1366, 1371, 1374, 1378, 1383, 1387, 1390,
 1394, 1406, 1411, 1414, 1418, 1423, 1427, 1430, 1434,
 1445, 1450, 1453, 1457, 1462, 1466, 1469, 1473, 1485,
 1499, 1532, 1534, 1539, 1541, 1543, 1547, 1551, 1553,
 1555, 1586, 1599, 1613, 1667, 1694, 3910
 \g__enumext_minipage_center_vii_bool . 4528,
 4545
 \g__enumext_minipage_center_viii_bool 4589,
 4605
 \g__enumext_minipage_center_X_bool . . . 173
 \l__enumext_minipage_hsep_v_dim 3850
 \l__enumext_minipage_hsep_vii_dim 4493
 \l__enumext_minipage_hsep_viii_dim . . . 4558
 \l__enumext_minipage_left_skip 83, 1486, 1564,
 1569, 1573, 1587, 1591, 1605, 1623, 1629
 \l__enumext_minipage_left_v_dim . . 3848, 3855
 \l__enumext_minipage_left_vii_dim 4489, 4501
 \l__enumext_minipage_left_viii_dim 4554, 4566
 \l__enumext_minipage_left_X_dim 83
 \g__enumext_minipage_right_skip 83, 1565, 1570,
 1574, 4527, 4588
 \l__enumext_minipage_right_skip . 54, 83, 1313,
 1319, 1324, 1326, 1328, 1487, 1488, 1494, 1499, 1500,
 1501, 1506, 1588, 1595, 1609, 1673, 1700
 \l__enumext_minipage_right_v_dim . 1689, 1698,
 3846, 3850
 \g__enumext_minipage_right_vii_dim 116, 4497,
 4524, 4547
 \l__enumext_minipage_right_vii_dim 116, 4487,
 4492, 4498
 \g__enumext_minipage_right_viii_dim . . 4562,
 4587, 4607
 \l__enumext_minipage_right_viii_dim . . 4552,
 4557, 4563
 \g__enumext_minipage_right_X_dim 173
 \g__enumext_minipage_right_X_skip 173
 __enumext_minipage_set_skip: . 54, 1311, 1311,
 1339
 \g__enumext_minipage_stat_int . . 102, 83, 1678,
 1705, 3753, 3764, 3769, 3853, 3902, 3907
 \l__enumext_minipage_temp_skip 83, 1385, 1395,
 1398, 1425, 1435, 1438, 1464, 1474, 1477, 1549, 1556,
 1558
 \l__enumext_miniright_code_vii_box 4535, 4539
 \g__enumext_miniright_code_vii_tl 117, 4530,
 4537, 4546
 \l__enumext_miniright_code_viii_box . . 4596,
 4600
 \g__enumext_miniright_code_viii_tl 4591, 4598,
 4606
 \l__enumext_miniright_code_X_box 173
 \l__enumext_mode_box_bool 636, 3304, 3477
 __enumext_multi_addvspace: 53, 101, 1266, 1266,
 3715
 __enumext_multi_set_vskip: 52, 1221, 1221, 1268
 \l__enumext_multicols_above_ii_skip . . 1240
 \l__enumext_multicols_above_iii_skip . . 1249
 \l__enumext_multicols_above_iv_skip . . 1258
 \l__enumext_multicols_above_v_skip 1285, 1299,
 1309, 1500
 \l__enumext_multicols_above_X_skip 75
 \l__enumext_multicols_below_ii_skip . . 1367,
 1376, 1380, 1392, 1397
 \l__enumext_multicols_below_iii_skip . 1407,
 1416, 1420, 1432, 1437
 \l__enumext_multicols_below_iv_skip . . 1446,
 1455, 1459, 1471, 1476

`\l__enumext_multicols_below_v_skip` 1289, 1303, 1501, 1535, 1542, 1544, 1554, 1557, 3892
`\l__enumext_multicols_below_X_skip` 75
`\g__enumext_multicols_right_X_skip` 75
`__enumext_multicols_start:` 101, 102, 3691, 3691, 3758
`__enumext_multicols_stop:` 102, 1663, 3721, 3721, 3774
`__enumext_nested_base_line_fix:` 47, 100, 973, 979, 3646
`__enumext_newlabel:nn` 31, 36, 79, 419, 419, 2582, 3100
`\l__enumext_newlabel_arg_one_tl` 31, 36, 79, 88, 159, 2575, 2583, 2645, 3089, 3101, 3139
`\l__enumext_newlabel_arg_two_tl` 31, 36, 78, 159, 2531, 2541, 2554, 2569, 2584, 3076, 3081, 3086, 3102
`__enumext_parse_foreach_keys:n` . . 5376, 5392, 5409
`__enumext_parse_foreach_keys:nn` . 5376, 5399, 5411
`__enumext_parse_keys:n` 47, 64, 3637, 3637, 3799
`__enumext_parse_keys_vii:n` 64, 4612, 4650, 4650
`__enumext_parse_keys_viii:n` . 4875, 4920, 4920
`__enumext_parse_save_key:n` 75, 2375, 2380, 2380
`__enumext_parse_save_key_vii:n` 75, 2370, 2380, 2388
`__enumext_parse_series:n` . . 64, 100, 119, 1863, 1863, 3645, 4656
`__enumext_parse_store_keys:n` 100
`\l__enumext_parsep_i_skip` 1238, 1242
`\l__enumext_parsep_ii_skip` 1247, 1251
`\l__enumext_parsep_iii_skip` 1256, 1260
`\l__enumext_parsep_vii_skip` 4828
`\l__enumext_parsep_viii_skip` 5079
`\l__enumext_partopsep_v_skip` . 1301, 1305, 1496, 1519
`\l__enumext_partopsep_viii_skip` 1597
`__enumext_phantomsection:` 36, 384, 412, 416, 432
`__enumext_pre_itemsep_skip:` . . 55, 1329, 1358, 1358
`__enumext_print_footnote:` . . 434, 456, 520, 525
`__enumext_print_footnote_mini:` 434, 486, 547, 552
`__enumext_print_footnote_standar:` 498, 514, 578
`__enumext_print_footnote_starred:` 498, 543, 558, 562
`__enumext_print_keyans_box:NN` 77, 2499, 2499, 2512, 2656, 2660, 3186, 3226, 5021, 5036
`\l__enumext_print_keyans_i_tl` 5160, 5182
`\l__enumext_print_keyans_ii_tl` 5164, 5183
`\l__enumext_print_keyans_iii_tl` 5168, 5184
`\l__enumext_print_keyans_iv_tl` 5172, 5185
`\l__enumext_print_keyans_star_bool` 47, 48, 130, 126, 985, 993, 5205, 5210
`\l__enumext_print_keyans_starred_tl` 129, 130, 126, 5156, 5203
`\l__enumext_print_keyans_vii_tl` 129, 5176, 5186
`\l__enumext_print_keyans_X_tl` 126
`__enumext_printkeyans:nnn` 130, 5179, 5187, 5190
`__enumext_redefine_item:` . 92, 3286, 3286, 3578
`\l__enumext_ref_key_arg_tl` 43, 46, 228, 737, 738, 750, 781, 784, 794, 800, 810, 851, 852, 862
`\l__enumext_ref_the_count_tl` . 43, 46, 744, 747, 750, 789, 791, 794, 805, 807, 810, 857, 859, 862
`__enumext_regex_counter_style:` . . 32, 43, 223, 223, 745, 790, 806, 858
`__enumext_register_counter_style:Nn` . . 600, 600, 605, 606, 607, 608, 609
`__enumext_remove_extra_parsep_vii:` . . 4630, 4852, 4852
`__enumext_remove_extra_parsep_viii:` . 4892, 5107, 5107
`__enumext_renew_footnote:` . . 434, 438, 504, 509
`__enumext_renew_footnote_mini:` 434, 468, 534, 539
`__enumext_renew_footnote_standar:` 498, 498, 570
`__enumext_renew_footnote_starred:` 498, 530, 4822, 5073
`\l__enumext_renew_the_count_v_tl` 860, 868, 870
`\l__enumext_renew_the_count_vii_tl` 792, 819, 821
`\l__enumext_renew_the_count_viii_tl` 808, 826, 828
`\l__enumext_renew_the_count_X_tl` 46
`__enumext_rescan_anskey_env:n` . . 85, 86, 2828, 2894, 2989, 2997
`__enumext_reset_global_bool:` . . 334, 337, 346
`__enumext_reset_global_int:` . . . 334, 336, 340
`__enumext_reset_global_tl:` 334, 338, 352
`__enumext_reset_global_vars:` . 34, 87, 334, 334, 3027
`\l__enumext_resume_active_bool` 64, 66, 57, 1867, 1987
`__enumext_resume_counter:` . . 66, 67, 1985, 1991, 1998
`__enumext_resume_counter:n` . 64, 66, 1956, 1961, 1985, 1985, 2055, 2063
`__enumext_resume_counter_save_ans:` 67, 1985, 1996, 2028
`__enumext_resume_counter_series:` . 67, 1985, 1994, 2011
`\g__enumext_resume_int` . . . 57, 1908, 2002, 2003
`__enumext_resume_last:n` 64, 65, 1863, 1869, 1882
`\l__enumext_resume_name_tl` 57, 1904, 1912, 1915, 1931, 1939, 1942, 1988, 1989, 2017, 2024
`__enumext_resume_save_counter:` . 65, 103, 119, 1895, 1895, 3780, 4674
`__enumext_resume_series:n` . 66, 1831, 1952, 1952
`__enumext_resume_starred:` . 68, 1832, 2049, 2049
`\g__enumext_resume_vii_int` 57, 1935, 2007, 2008
`\l__enumext_rightmargin_vii_dim` . . 4343, 4347, 4352
`\l__enumext_rightmargin_viii_dim` . 4374, 4378, 4383
`__enumext_safe_exec:` . . 39, 100, 3626, 3626, 3798
`__enumext_safe_exec_vii:` . 39, 4611, 4633, 4633
`__enumext_safe_exec_viii:` 124, 4874, 4896, 4896
`__enumext_second_part:` . . 102, 3760, 3760, 3812
`__enumext_second_part_v:` . . . 3842, 3898, 3946
`\l__enumext_series_name_tl` 66, 67
`\l__enumext_series_str` . 65, 100, 119, 1829, 1865, 1873, 1874, 1876, 1878, 1899, 1902, 1906, 1926, 1929, 1933, 3641, 4654
`__enumext_set_error:nn` 5286, 5323, 5325
`__enumext_set_item_width:` 103, 3782, 3782, 3808
`__enumext_set_parse:n` 5286, 5297, 5313

`\l__enumext_setkey_tmpa_int` . . . [117](#), [5290](#), [5294](#)
`\l__enumext_setkey_tmpa_seq` . . . [117](#), [5288](#), [5298](#),
[5304](#), [5306](#), [5308](#), [5320](#)
`\l__enumext_setkey_tmpa_tl` [117](#), [5296](#), [5300](#)
`\l__enumext_setkey_tmpb_seq` . . . [117](#), [5289](#), [5292](#),
[5296](#), [5297](#)
`\l__enumext_setkey_tmpb_tl` [117](#), [5315](#), [5317](#), [5318](#)
`\l__enumext_show_answer_bool` . [2315](#), [2339](#), [2668](#),
[3156](#), [3170](#), [4233](#), [5019](#)
`__enumext_show_length:nnn` . . [50](#), [231](#), [231](#), [5534](#),
[5535](#), [5536](#), [5537](#), [5538](#), [5539](#), [5540](#), [5541](#), [5542](#), [5543](#),
[5549](#), [5550](#), [5551](#), [5552](#), [5553](#), [5554](#), [5555](#), [5556](#), [5557](#),
[5558](#)
`\l__enumext_show_position_bool` . . . [2318](#), [2342](#),
[2672](#), [3160](#), [3171](#), [4234](#), [5023](#)
`\g__enumext_standar_bool` [33](#), [100](#), [30](#), [253](#), [256](#), [275](#),
[349](#), [500](#), [516](#), [1897](#), [1962](#), [1974](#), [2000](#), [2013](#), [2051](#),
[2191](#), [2205](#), [2536](#), [2549](#), [2564](#), [3661](#)
`\l__enumext_standar_bool` [100](#), [103](#), [30](#), [1647](#), [2537](#),
[3633](#), [3779](#), [4647](#)
`\l__enumext_standar_first_bool` [33](#), [100](#), [30](#), [280](#),
[1884](#), [2031](#), [2093](#), [2100](#)
`__enumext_standar_item_vii:w` . [120](#), [121](#), [4704](#),
[4722](#), [4724](#)
`__enumext_standar_item_viii:w` [126](#), [4947](#), [4965](#),
[4967](#)
`__enumext_standar_ref:` [43](#), [735](#), [754](#), [3580](#)
`__enumext_standar_ref:n` [43](#), [727](#), [735](#), [735](#)
`\g__enumext_standar_series_tl` . [57](#), [1886](#), [1887](#),
[2053](#), [2056](#)
`__enumext_standar_unknown_keys:n` [3407](#), [3411](#),
[3415](#)
`__enumext_standar_unknown_keys:nn` [3407](#), [3417](#),
[3419](#)
`\g__enumext_starred_bool` [33](#), [118](#), [30](#), [263](#), [266](#), [290](#),
[350](#), [1646](#), [1924](#), [1967](#), [1978](#), [2005](#), [2020](#), [2059](#), [2165](#),
[2211](#), [2527](#), [3070](#), [4548](#)
`\l__enumext_starred_bool` [119](#), [124](#), [30](#), [2565](#), [2600](#),
[2606](#), [2654](#), [2943](#), [2948](#), [3179](#), [3192](#), [3634](#), [4646](#), [4673](#),
[4908](#), [4912](#)
`__enumext_starred_columns_set_vii:` . . [4325](#),
[4325](#), [4621](#)
`__enumext_starred_columns_set_viii:` . [4325](#),
[4356](#), [4883](#)
`\l__enumext_starred_first_bool` [33](#), [119](#), [30](#), [295](#),
[983](#), [992](#), [1889](#), [2040](#), [2093](#), [2100](#)
`__enumext_starred_item:nn` . . . [3249](#), [3249](#), [3292](#)
`__enumext_starred_item_exec:` . [127](#), [4981](#), [5011](#),
[5051](#)
`__enumext_starred_item_vii:w` . [120](#), [121](#), [4704](#),
[4721](#), [4738](#)
`__enumext_starred_item_vii_aux_i:w` . . [4704](#),
[4743](#), [4746](#)
`__enumext_starred_item_vii_aux_ii:w` . [4704](#),
[4744](#), [4749](#), [4751](#)
`__enumext_starred_item_vii_aux_iii:w` [4704](#),
[4754](#), [4761](#)
`__enumext_starred_item_viii:w` [126](#), [4964](#), [4981](#),
[4981](#)
`__enumext_starred_item_viii_aux_i:w` . . [126](#),
[4981](#), [4986](#), [4989](#)
`__enumext_starred_item_viii_aux_ii:w` . [126](#),
[4981](#), [4987](#), [5004](#), [5006](#)
`__enumext_starred_joined_item_vii:n` [114](#), [120](#),
[4387](#), [4387](#), [4719](#)

`__enumext_starred_joined_item_viii:n` . [114](#),
[126](#), [4387](#), [4436](#), [4962](#)
`__enumext_starred_ref:` [44](#), [779](#), [815](#), [3611](#)
`__enumext_starred_ref:n` [44](#), [773](#), [779](#), [779](#)
`\g__enumext_starred_series_tl` . [57](#), [1891](#), [1892](#),
[2061](#), [2064](#)
`__enumext_starred_unknown_keys:n` [3389](#), [3391](#),
[3393](#)
`__enumext_starred_unknown_keys:nn` [3389](#), [3395](#),
[3397](#)
`__enumext_start_from:NNn` [45](#), [873](#), [873](#), [886](#), [908](#),
[914](#)
`\l__enumext_start_i_int` [2003](#), [2015](#), [2034](#)
`__enumext_start_item_tmp_vii:` [118](#), [4624](#), [4704](#),
[4704](#)
`__enumext_start_item_tmp_viii:` . . [4886](#), [4947](#),
[4947](#)
`__enumext_start_item_vii:w` [121](#), [122](#), [4730](#), [4735](#),
[4758](#), [4765](#), [4813](#), [4813](#)
`__enumext_start_item_viii:w` . . [126](#), [4973](#), [4978](#),
[5009](#), [5064](#), [5064](#)
`\g__enumext_start_line_tl` [33](#), [30](#), [283](#), [298](#), [355](#),
[2235](#), [2240](#), [2245](#), [2259](#), [2264](#), [2269](#)
`__enumext_start_list:nn` . [35](#), [97](#), [373](#), [375](#), [3802](#),
[3935](#), [4615](#), [4877](#)
`__enumext_start_list_tag:n` . . [3948](#), [3974](#), [4823](#),
[5074](#)
`__enumext_start_mini_vii:` [119](#), [4485](#), [4485](#), [4665](#)
`__enumext_start_mini_viii:` . . . [125](#), [4550](#), [4550](#),
[4931](#)
`__enumext_start_save_ans_msg:` . . [68](#), [69](#), [2077](#),
[2077](#), [2102](#)
`__enumext_start_store_level:` . [100](#), [3655](#), [3655](#),
[3801](#)
`__enumext_start_store_level_vii:` [120](#), [4614](#),
[4676](#), [4676](#)
`\l__enumext_start_vii_int` . . . [2008](#), [2022](#), [2043](#)
`\l__enumext_start_X_int` [96](#)
`__enumext_stop_item_tmp_vii:` . . [118](#), [120](#), [122](#),
[4623](#), [4629](#), [4706](#), [4815](#)
`__enumext_stop_item_tmp_viii:` [125](#), [4885](#), [4891](#),
[4949](#), [5066](#)
`__enumext_stop_item_vii:` [122](#), [123](#), [4813](#), [4815](#),
[4835](#)
`__enumext_stop_item_viii:` . . . [5064](#), [5066](#), [5090](#)
`__enumext_stop_list:` [35](#), [116](#), [119](#), [373](#), [376](#), [3726](#),
[3734](#), [3888](#), [3895](#), [4508](#), [4516](#), [4573](#), [4580](#)
`__enumext_stop_list_tag:n` . . . [3948](#), [3990](#), [4838](#),
[5093](#)
`__enumext_stop_mini_vii:` [116](#), [119](#), [4485](#), [4504](#),
[4669](#)
`__enumext_stop_mini_viii:` [125](#), [4550](#), [4569](#), [4935](#)
`__enumext_stop_save_ans_msg:` . [68](#), [2077](#), [2082](#),
[3016](#)
`__enumext_stop_start_list_tag:` . . [3948](#), [3982](#),
[4825](#), [5076](#)
`__enumext_stop_store_level:` . . [101](#), [102](#), [3684](#),
[3684](#), [3727](#), [3735](#)
`__enumext_stop_store_level_vii:` [116](#), [119](#), [120](#),
[4509](#), [4517](#), [4676](#), [4686](#)
`\l__enumext_store_active_bool` [30](#), [69](#), [108](#), [2032](#),
[2041](#), [2109](#), [2741](#), [3659](#), [3672](#), [3820](#), [3828](#), [4186](#), [4678](#),
[4688](#), [4898](#), [4914](#)
`__enumext_store_active_keys:n` [74](#), [75](#), [100](#), [2348](#),

2348, 3652
 __enumext_store_active_keys_vii:n 74, 75, 119, 2348, 2358, 4657
 __enumext_store_addto_prop:n 76, 88, 2423, 2423, 2431, 2591, 3051, 5014
 __enumext_store_addto_seq:n 76, 89, 2432, 2432, 2436, 2443, 2457, 2465, 2474, 2488, 2496, 2649, 3144
 \l__enumext_store_anskey_arg_tl 30, 79, 80, 108, 2597, 2602, 2604, 2609, 2616, 2619, 2629, 2634, 2637, 2643, 2649
 __enumext_store_anskey_code:n 79, 82, 86, 2588, 2588, 2734, 2987, 2995
 \l__enumext_store_anskey_env_tl .. 30, 85, 108, 2917, 2921, 2927, 2989, 2997
 \l__enumext_store_anskey_opt_tl .. 30, 86, 108, 2918, 2945, 2951, 2958, 2964, 2974, 2984, 2993
 __enumext_store_anskey_safe_outer: 82
 \g__enumext_store_columns_break_bool . 2841, 2942, 3004
 \l__enumext_store_columns_break_bool . 2599, 2690
 \l__enumext_store_current_label_tl 30, 88, 89, 126, 108, 3032, 3035, 3038, 3044, 3049, 3051, 3108, 3111, 3114, 3120, 3125, 3135, 3144, 4991, 4996, 5000, 5013, 5014, 5016
 \l__enumext_store_current_label_tmp_tl . 30, 108, 3443, 3448
 \l__enumext_store_current_opt_arg_tl 30, 126, 108, 3154, 3167, 3173, 5002
 __enumext_store_internal_ref: .. 78, 79, 2513, 2513, 2594
 \g__enumext_store_item_join_int .. 2844, 2949, 2953, 3005
 \l__enumext_store_item_join_int .. 2607, 2611, 2693
 \g__enumext_store_item_star_bool . 2846, 2956, 3006
 \l__enumext_store_item_star_bool . 2614, 2695
 \g__enumext_store_item_symbol_sep_dim 2851, 2971, 2976, 3008
 \l__enumext_store_item_symbol_sep_dim 2626, 2631, 2700
 \g__enumext_store_item_symbol_tl . 2849, 2962, 2966, 3007
 \l__enumext_store_item_symbol_tl . 2617, 2621, 2698
 \l__enumext_store_keyans_item_opt_sep_tl 2301, 3042, 3046, 3118, 3122, 4994, 4998
 __enumext_store_level_close: . 76, 2437, 2461, 3688
 __enumext_store_level_close_vii: . 77, 2468, 2492, 4692
 __enumext_store_level_open: 76, 101, 2437, 2437, 3667, 3680
 __enumext_store_level_open_vii: .. 77, 2468, 2468, 4682
 \g__enumext_store_name_tl 30, 69, 108, 354, 361, 362, 363, 364, 2085, 2111, 2234, 2239, 2244, 2258, 2263, 2268, 3014
 \l__enumext_store_name_tl 30, 69, 70, 108, 1918, 1921, 1945, 1948, 2036, 2045, 2080, 2089, 2090, 2111, 2112, 2113, 2115, 2116, 2118, 2120, 2121, 2123, 2125, 2126, 2150, 2425, 2427, 2434, 2577, 2578, 2680, 2923, 3091, 3092, 3205, 3218, 5031
 \l__enumext_store_ref_key_bool 79, 2324, 2592, 2640, 3055, 3132
 \l__enumext_store_save_key_vii_bool .. 2360, 2390
 \l__enumext_store_save_key_vii_tl 2362, 2363, 2391, 2392, 2472, 2480, 2484, 2488
 \l__enumext_store_save_key_X_bool .. 74, 126
 \l__enumext_store_save_key_X_tl .. 74, 75, 126
 \l__enumext_store_upper_level_X_bool .. 126
 __enumext_storing_exec: 69, 84, 2087, 2103, 2107
 __enumext_storing_set:n 68, 69, 2072, 2087, 2087
 \l__enumext_the_counter_v_tl 859
 \l__enumext_the_counter_vii_tl 791
 \l__enumext_the_counter_viii_tl 807
 \l__enumext_the_counter_X_tl 46
 __enumext_tmp:n 41, 45, 50, 56, 67, 74, 75, 82, 90, 95, 96, 107, 130, 137, 162, 166, 173, 193, 632, 641, 1825, 1836, 2068, 2076, 2129, 2147, 2288, 2329, 2330, 2347, 2366, 2379, 2515, 2522, 2523, 2544, 2557, 2560, 2571, 3057, 3064, 3367, 3374, 3407, 3414, 3550, 3590, 3591, 3625
 __enumext_tmp:nn 642, 663, 664, 698, 699, 714, 903, 928, 1005, 1027, 1028, 1048, 1102, 1110, 1111, 1125, 1190, 1206, 1207, 1220, 1714, 1730, 3351, 3366
 __enumext_tmp:nnn 715, 731, 732, 733, 734, 761, 777, 778
 __enumext_tmp:nnnnn 929, 954, 957, 960, 962, 964, 967, 970
 __enumext_tmp:w 5135, 5138
 \l__enumext_tmpa_vii_int 4335, 4338, 4347, 4378
 \l__enumext_tmpa_viii_int 4366, 4369
 \l__enumext_tmpa_X_dim 173
 \l__enumext_tmpa_X_int 173
 \l__enumext_topsep_v_skip 1287, 1291, 1490, 4179
 \l__enumext_topsep_vii_skip .. 1567, 1576, 1580
 \l__enumext_topsep_viii_skip . 1589, 1611, 1615
 __enumext_undefine_anskey_env: . 83, 87, 2774, 2774, 3025
 __enumext_unskip_unkern: .. 33, 237, 237, 1340, 1512, 3729, 3730, 3770, 3890, 3891, 3908, 4829, 4830, 5080, 5081
 \l__enumext_vspace_a_star_v_bool 1763
 \l__enumext_vspace_a_star_vii_bool . . . 1785
 \l__enumext_vspace_a_star_viii_bool . . . 1796
 \l__enumext_vspace_a_star_X_bool 96
 __enumext_vspace_above: 62, 102, 1731, 1731, 3740
 __enumext_vspace_above_v: . 62, 1759, 1759, 3844
 \l__enumext_vspace_above_v_skip .. 1761, 1765, 1767
 __enumext_vspace_above_vii: 63, 119, 1781, 1781, 4662
 \l__enumext_vspace_above_vii_skip 1783, 1787, 1789
 __enumext_vspace_above_viii: . 63, 1781, 1792, 4929
 \l__enumext_vspace_above_viii_skip 1794, 1798, 1800
 \l__enumext_vspace_b_star_v_bool 1774
 \l__enumext_vspace_b_star_vii_bool . . . 1807
 \l__enumext_vspace_b_star_viii_bool . . . 1818
 \l__enumext_vspace_b_star_X_bool 96
 __enumext_vspace_below: 62, 103, 1745, 1745, 3778
 __enumext_vspace_below_v: . 62, 1770, 1770, 3917
 \l__enumext_vspace_below_v_skip .. 1772, 1776, 1778

- _enumext_vspace_below_vii: 63, 119, 1803, 1803, 4672
- \l_enumext_vspace_below_vii_skip 1805, 1809, 1811
- _enumext_vspace_below_viii: . 63, 1803, 1814, 4937
- \l_enumext_vspace_below_viii_skip 1816, 1820, 1822
- _enumext_widest_from:nNNn . . 45, 887, 887, 902, 921
- \g_enumext_widest_label_tl 28, 40, 63, 620, 624, 628
- \l_enumext_wrap_label_opt_v_bool 3437
- \l_enumext_wrap_label_opt_vii_bool 121, 4729
- \l_enumext_wrap_label_opt_viii_bool . . 126, 4972
- \l_enumext_wrap_label_opt_X_bool 96
- \l_enumext_wrap_label_v_bool 3433, 3437, 3445, 3492, 3509
- \l_enumext_wrap_label_vii_bool . . 121, 4729, 4733, 4741, 4805
- \l_enumext_wrap_label_viii_bool . 126, 4972, 4976, 4984, 5056
- \l_enumext_wrap_label_X_bool 96
- _enumext_wrapper_label_v:n . 3494, 3511, 4242
- _enumext_wrapper_label_vii:n 4807
- _enumext_wrapper_label_viii:n 5058
- \l_enumext_write_aux_file_tl . 31, 79, 89, 159, 2580, 2586, 3098, 3104
- enumext* 5, 4609
- enumXi 583
- enumXii 583
- enumXiii 583
- enumXiv 583
- enumXv 583
- enumXvi 583
- enumXvii 583
- enumXviii 583
- Environments provide by **enumext**:
 - anskey* 30, 69, 75, 78, 80, 83–85, 87, 100, 101, 120, 129, 130, 135, 137
 - enumext* 27, 28, 31–33, 37–41, 43, 44, 46, 48–52, 58, 59, 63–66, 68–71, 73–79, 81, 83, 86–88, 94, 95, 99–101, 106, 113, 114, 116, 117, 120, 122, 124, 125, 127–132, 136, 139, 140
 - enumext 27, 28, 32, 33, 37–54, 57, 60–66, 68–71, 73–76, 78, 79, 81, 83, 86–88, 91–95, 97, 98, 101, 103, 104, 108, 113, 116, 118, 120, 122, 124, 129–132, 136, 137, 139
 - keyans* 27, 28, 30–34, 37–40, 43–46, 48–52, 58, 59, 63, 69, 70, 73, 74, 76, 83, 88, 94, 99, 106, 114, 115, 124, 125, 136, 138, 140
 - keyanspic . . 27, 28, 30, 31, 34, 40, 44, 69, 70, 73, 76, 83, 88–90, 94, 106–112, 138
 - keyans 27, 28, 30, 31, 33, 34, 37, 38, 40, 41, 44, 46, 48–51, 54, 57, 60–62, 69, 70, 73, 74, 76, 83, 88–90, 94–98, 104, 106, 108, 109, 112, 116, 125, 136, 138
- Environments:
 - center 113
 - description 93, 113
 - enumerate 113
 - flushleft 113
 - flushright 113
 - itemize 113
 - list . 32, 35, 81, 93, 97, 102, 103, 106, 108–110, 113, 116
- lrbox 122
- minipage 32, 35–37, 39, 52, 54, 55, 108, 111–113, 116, 117, 123
- multicols 52–55, 60, 101, 102
- quotation 113
- quote 113
- scontents 84, 86
- tabbing 113
- trivlist 113
- verbatim 113
- verse 113
- exp commands:
 - \exp_after:wN 5138
 - \exp_args:Ne 2986, 2994, 3649, 5126
 - \exp_args:NV . . . 2706, 2861, 3377, 3395, 3417, 5411
 - \exp_not:N 54, 623, 750, 794, 810, 862, 1058, 1061, 1072, 1073, 1074, 1085, 1086, 1097, 1098, 2645, 2677, 2678, 3137, 3202, 3203, 3215, 3216, 5028, 5029, 5135
 - \exp_not:n 285, 300, 313, 321, 329, 689, 709, 750, 794, 810, 862, 1059, 1852, 1861, 2312, 2409, 2421, 2583, 2611, 2621, 2631, 2645, 2646, 2953, 2966, 2976, 3101, 3139, 3141, 4061, 5240, 5250, 5443, 5448
- F**
 - \fbox 2295
 - \fboxrule 2295
 - \fboxsep 2295
 - file commands:
 - \file_input_stop: 5847
 - first 1111
 - font 642
 - \footnote 37
 - \footnote 37, 440, 470
 - \footnotemark 450, 480
 - \footnotesize 2678, 3203, 3216, 5029
 - \footnotetext 436
 - \foreachkeyans 17, 133, 5376
- G**
 - \getkeyans 17, 129, 5124
 - group commands:
 - \group_begin: . . 2676, 2721, 2896, 2983, 3201, 3214, 5027, 5181
 - \group_end: 2683, 2737, 3000, 3208, 3221, 5034, 5188
- H**
 - \hbadness 4840, 5095
 - hbox commands:
 - \hbox_overlap_left:n 3282, 4798
 - \hbox_set:Nn 612, 4107
 - \hbox_set_end: 4839, 5094
 - \hbox_set_to_wd:Nnw 4816, 5067
 - \hfill 672, 677, 683, 684, 1670, 1697, 2645, 3137, 4512, 4576
 - hook commands:
 - \hook_gput_code:nnn 5, 203, 207, 211, 384
 - \hook_gremove_code:nn 86, 2912
 - \hook_gset_rule:nnnn 385
 - \hook_if_empty:nTF 2910
 - \hyperlink 80, 89
 - \hyperlink 2645, 3137
 - \hypertarget 36
 - \hypertarget 411
- I**
 - \IfDocumentMetadataTF . . 502, 518, 532, 545, 3299, 3472, 3976, 3984, 3992, 4028, 4036, 4044, 4130, 4139, 4147,

4154, 4159, 4207, 4216, 4300, 4308, 4510, 4574, 4620, 4628, 4774, 4882, 4890

\IfHyperBoolean 392

\IfPackageLoadedTF 7, 15, 388, 401

\ignorespaces 1061, 1074, 1086, 1098, 4625, 4702, 4735, 4758, 4765, 4811, 4831, 4887, 4945, 4978, 5009, 5062, 5082

\inputlineno 285, 300, 313, 321, 329

int commands:

\int_add:Nn 4420, 4469

\int_case:nn 1235, 1360, 2160, 2186, 2225, 2249

\int_case:nnTF 239

\int_compare:nNnTF 566, 782, 798, 817, 824, 1330, 1349, 1503, 1521, 1633, 1652, 1664, 1692, 2273, 2279, 2745, 2749, 2753, 2761, 2807, 2811, 2815, 3012, 3033, 3074, 3079, 3084, 3109, 3197, 3631, 3642, 3664, 3677, 3693, 3708, 3723, 3764, 3829, 3833, 3861, 3886, 3902, 4079, 4190, 4194, 4390, 4400, 4416, 4439, 4449, 4465, 4638, 4642, 4680, 4690, 4842, 4854, 4903, 4915, 5097, 5109, 5294, 5426

\int_compare_p:nNn 254, 264, 276, 277, 291, 292, 1639, 1640, 2166, 2192, 2528, 2538, 2550, 2551, 2566, 2607, 2784, 2785, 2796, 2797, 2949, 3674

\int_decr:N 4419, 4468

\int_eval:n 371, 916, 2427, 2578, 2678, 3092, 3203, 3216, 3565, 3610, 4408, 4457, 5029

\int_from_alph:n 881, 895

\int_from_roman:n 883, 897

\int_gadd:Nn 4421, 4470

\int_gdecr:N 2169, 2174, 2178, 2182, 2195

\int_gincr:N 2002, 2007, 2590, 3147, 3236, 3270, 3452, 3753, 3853, 4231, 4708, 4784, 4951, 5018

\int_gset:Nn 448, 478, 2218

\int_gset_eq:NN 445, 475, 1901, 1908, 1914, 1920, 1928, 1935, 1941, 1947

\int_gzero:N 342, 343, 344, 1678, 1705, 2285, 3005, 3769, 3907, 4865, 5121

\int_if_exist:NTF 1876, 1912, 1918, 1939, 1945, 2123

\int_incr:N 2760, 3630, 3824, 4078, 4637, 4707, 4902, 4950

\int_mod:nn 4856, 5111

\int_new:N 24, 25, 26, 27, 28, 29, 57, 58, 83, 100, 119, 140, 141, 152, 153, 154, 156, 167, 168, 176, 177, 178, 179, 180, 1878, 2126

\int_set:Nn 877, 881, 883, 2015, 2022, 2034, 2043, 2897, 4294, 4295, 4335, 4366, 4389, 4395, 4411, 4438, 4444, 4460, 4840, 5095, 5290, 5428

\int_set_eq:NN 2003, 2008, 4418, 4467

\int_sign:n 2220

\int_step_function:nnN 2544, 2557, 2571

\int_step_function:nnnN 5432

\int_step_inline:nn 5342

\int_step_inline:nnn 4296

\int_to_roman:n 215, 2524, 2561

\int_use:N 364, 369, 370, 1331, 1350, 1665, 2017, 2024, 2036, 2045, 3565, 3585, 3610, 3650, 3694, 3703, 3718, 3724, 4393, 4394, 4406, 4442, 4443, 4455, 5763, 5767, 5773, 5777

\int_zero:N 4846, 5101

\item 91, 95, 120, 122, 125, 128, 377, 2445, 2451, 2476, 2482, 2604, 3111, 3114, 3288, 3456, 4134, 4135, 4622, 4624, 4884, 4886, 5016

\item* 5, 15, 73, 3454

item-pos* 3351

item-sym* 3351

\itemindent 98

\itemindent 97

itemindent 1005

\itemsep 4123

\itemwidth 582, 2295, 3784, 3790, 3921, 3927, 4429, 4433, 4478, 4482

K

keyans 14, 3930

keyans* 14, 4872

keyanspic 15, 4125

Keys for \anskey provide by **enumext**:

break-col 79, 81, 84–86

item-join 79, 81, 84–86

item-pos* 80, 81, 84–86

item-star 80, 81, 84–86

item-sym* 80, 81, 84–86

Keys for anskey* provide by **enumext**:

break-col 79, 81, 84–86

item-join 79, 81, 84–86

item-pos* 80, 81, 84–86

item-star 80, 81, 84–86

item-sym* 80, 81, 84–86

Keys for environments provide by **enumext**:

above* 29, 48, 61–63, 102, 119

above 29, 48, 61–63, 102, 119, 125

after 50, 103, 119, 125

align 29, 41, 91, 93, 96, 121, 135

base-fix 47, 64, 75, 100

before* 50, 102, 119, 125

before 50

below* 29, 61–63, 103, 119

below 29, 61–63, 103, 119, 125

check-ans 31–33, 68–73, 76, 87, 90, 103, 104, 119, 124, 137

columns-sep 51, 101, 123

columns 29, 51, 61, 101

first 50, 123

font 41, 93, 96, 111, 121

item-pos* 92, 94

item-sym* 30, 92, 94

itemindent 29, 48, 49, 91, 92, 95, 96, 123

itemsep 46, 99, 123

label-pos 108, 109, 111, 112

label-sep 108

labelsep 41, 98, 121

labelwidth 40–44, 46, 98, 121

label 28, 40, 42, 45, 46, 109, 113

layout-sep 108

layout-sty 108, 112, 113

layout-top 108

lisparindent 99

list-indent 29, 48, 109

list-offset 48, 103, 105

listparindent 48, 123

mark-ans 73, 76, 80

mark-pos 73, 74, 135

mark-ref 73, 76, 78, 80

mini-env 29, 37–39, 51, 60, 61, 76, 102, 113, 116, 117, 119, 125

mini-right* 29, 32, 52, 76, 117, 119

mini-right 29, 32, 52, 59, 76, 117, 119

mini-sep 29, 51, 76, 102

mode-box 41, 91, 93, 96, 97

no-store 31, 68–70, 75, 81, 91, 92

noitemsep	46
nosep	46
parindent	99
parsep	46, 99, 109, 123
partopsep	46
ref	28, 32, 42–44, 136
resume*	28, 63, 64, 68, 69, 75, 103, 119, 131
resume	28, 35, 63–69, 75, 76, 103, 119, 131
rightmargin	48, 114
save-ans	30, 35, 64–70, 72, 74–76, 81–84, 87–89, 95, 104, 111, 122, 124, 125, 127, 129, 131, 136
save-key	30, 64, 75, 100, 119
save-pos	76
save-ref	31, 36, 73, 76, 78–80, 88, 89, 96, 127
save-sep	73, 76, 88, 126
series	28, 63–68, 76, 100, 103, 119, 131
show-ans	73, 74, 76, 77, 79, 80, 96, 111, 127
show-length	33, 49, 136
show-pos	30, 73, 74, 77, 79, 80, 90, 96, 111, 127
start*	29, 45, 46, 64
start	29, 32, 45, 46, 64
store-key	74
topsep	46, 48, 109
widest	28, 32, 45, 46
wrap-ans	39, 73, 76, 77, 80
wrap-label*	29, 41, 91, 93, 95, 96, 121, 126
wrap-label	29, 41, 91–93, 95, 96, 109, 111, 121, 126
wrap-opt	73, 76, 96, 111
keys commands:	
\keys_define:nn	634, 644, 666, 701, 717, 763, 832, 905, 931, 973, 1007, 1030, 1104, 1113, 1192, 1209, 1716, 1827, 2070, 2131, 2290, 2332, 2368, 2373, 2688, 2839, 2875, 3353, 3369, 3389, 3409, 4050, 5152, 5252, 5368, 5376
\keys_if_exist_p:nn	5364, 5365
\l_keys_key_str	81, 84, 2706, 2861, 3377, 3395, 3417, 5411, 5519
\keys_precompile:nnN	130, 199, 199, 5154, 5158, 5162, 5166, 5170, 5174, 5394
\keys_set:nn	658, 999, 1215, 1721, 1726, 1964, 1969, 2056, 2064, 2726, 3644, 3649, 3840, 4068, 4071, 4089, 4655, 4924, 5256, 5261, 5262, 5263, 5264, 5267, 5272, 5273, 5274, 5275, 5276, 5277, 5278, 5310, 5420
\keys_set_known:nn	2993
keyval commands:	
\keyval_parse:NNn	1841, 2398, 5228
L	
label	715, 761, 832
label-pos	4050
label-sep	4050
Labels provide by <code>enumext</code> :	
\Alph*	40
\Roman*	40
\alph*	40
\arabic*	32, 40
\roman*	40
\labelsep	4119
labelsep	642
\labelwidth	40
\labelwidth	4119
labelwidth	642
\lastnodetype	239
layout-sep	4050
layout-sty	4050
layout-top	4050
\leftmargin	98
\leftmargin	97, 4119
legacy commands:	
\legacy_if:nTF	4769, 4772, 5041, 5044
\legacy_if_gset_false:n	572, 4525
\legacy_if_set_false:n	4771, 5043
\legacy_if_set_true:n	4734, 4757, 4764, 4778, 4977, 5008
\linewidth	102
\linewidth	3748, 3784, 3850, 3921, 4293, 4338, 4369, 4491, 4556
\list	375
list-indent	1005
list-offset	1005
\listparindent	4121
listparindent	1005
M	
\makebox	113
\makebox	2503, 2505, 3335, 3506, 4224, 4802, 5053
\makeLabel	91, 93, 96, 113
\makeLabel	91, 95, 3315, 3331, 3488, 3502
mark-ans	2288
mark-pos	2288, 2330
mark-ref	2288
mini-env	1190
mini-sep	1190
\minipage	381
\miniright	11, 60, 1631, 1682, 1709, 3767, 3905
mode commands:	
\mode_if_math:TF	2769, 2823
\mode_if_vertical:TF	1269, 1297, 1317, 1341, 1492, 1513
\mode_leave_vertical:	988, 995, 1058, 1072, 2501, 3280, 4796
mode-box	632
msg commands:	
\msg_error:nn	1684, 1711, 2730, 2763, 2767, 2821, 2929, 3831, 3835, 4081, 4137, 4192, 4640, 4905, 4917, 5279, 5338
\msg_error:nnn	740, 786, 802, 854, 1635, 1642, 1649, 1680, 1707, 1976, 1980, 2095, 2712, 2771, 2789, 2801, 2809, 2813, 2817, 2825, 2867, 3383, 3401, 3423, 4644, 4910, 5140, 5149, 5221, 5326, 5357, 5366, 5403, 5424
\msg_error:nnnn	2715, 2743, 2747, 2751, 2755, 2870, 3386, 3404, 3426, 3822, 4188, 4196, 4900, 5200, 5406
\msg_error:nnnnn	688, 708, 2311, 4060
\msg_fatal:nn	3632
\msg_fatal:nnn	586
\msg_info:nnn	9, 12, 17, 20, 390, 403
\msg_line_context:	5484, 5489, 5494, 5523, 5528, 5533, 5548, 5563, 5567, 5571, 5575, 5579, 5583, 5590, 5597, 5603, 5617, 5621, 5626, 5630, 5634, 5638, 5643, 5647, 5651, 5655, 5660, 5695, 5699, 5704, 5709, 5713, 5718, 5794, 5798, 5803, 5808, 5813, 5817, 5821, 5825, 5829, 5833, 5837, 5841, 5845
\msg_log:nnn	2115, 2120, 2125
\msg_log:nnnnn	368, 2258, 2263, 2268
\msg_log:nnnnnn	360
\msg_new:nnn	5451, 5455, 5459, 5463, 5468, 5481, 5486, 5491, 5496, 5505, 5513, 5517, 5521, 5526, 5531, 5546, 5561, 5565, 5569, 5573, 5577, 5581, 5585, 5594, 5600, 5606, 5610, 5614, 5619, 5624, 5628, 5632, 5636, 5641, 5645, 5649, 5653, 5658, 5693, 5697, 5702, 5707, 5711,

5716, 5792, 5796, 5801, 5806, 5811, 5815, 5819, 5823, 5827, 5831, 5835, 5839, 5843	\phantomsection 36
\msg_new:nnnn 5472, 5663, 5672, 5681, 5687, 5720, 5730, 5740, 5750, 5760, 5770, 5780, 5786	\phantomsection 412
\msg_term:nnnn 2079, 2084, 3574, 3584, 3616, 3621	prg commands:
\msg_term:nnnnn 2239	\prg_do_nothing: 416
\msg_warning:nn 3766, 3904	\prg_new_protected_conditional:Npnn 217
\msg_warning:nnnn 2276, 2282, 3522, 3527, 4392, 4405, 4441, 4454	\prg_replicate:nn 234
\msg_warning:nnnnn 2234, 2244	\prg_return_false: 221
\multicolsep 101	\prg_return_true: 220
\multicolsep 1334, 1506, 3714, 3877	\printkeyans 18, 129, 5179
N	prop commands:
\NeedsTeXFormat 3	\prop_const_from_keyval:Nn 5327
\NewCommandCopy 377	\prop_count:N 362, 2427, 2578, 2680, 3092, 3205, 3218, 5031, 5429
\newcounter 589	\prop_get:NnNTF 5353
\NewDocumentCommand 1631, 2718, 4184, 5124, 5179, 5286, 5335, 5413	\prop_gput_if_not_in:Nnn 2425
\NewDocumentEnvironment 3796, 3930, 4125, 4609, 4872	\prop_if_exist:NTF 2113, 5144, 5422
\newenvsc 2832	\prop_item:Nn 5146, 5446
\newlabel 37	\prop_new:N 2116
\newlabel 423	\ProvidesExplPackage 4
no-store 2129	R
\noindent 3755, 4500, 4565, 4845, 5100	\raggedcolumns 3717, 3880
\nointerlineskip 1343, 1346, 1515, 1518, 1672, 1699, 4500, 4565	\raisebox 4255
noitemsep 929	\ref 78, 88
\nopagebreak 1280, 1308, 1343, 1346, 1515, 1518, 1622, 1628	ref 715, 761, 832
\normalfont 2677, 3202, 3215, 5028	\refstepcounter 4781, 5046
nosep 929	regex commands:
P	\regex_match:nnTF 219, 880, 882, 894, 896, 2925
Packages:	\regex_replace_once:nnN 227
caption 117	\renewcommand 750, 794, 810, 862
enumext 27, 39, 42, 68, 93, 98, 108, 135	\RenewDocumentCommand 440, 470, 1682, 1709, 3288, 3315, 3331, 3456, 3488, 3502, 4135
enumitem 40	\RequirePackage 13, 21
expl3 113	resume 1825
footnotehyper 36, 38	resume* 1825
hyperref 31, 32, 36, 37, 80, 89, 122, 135	rightmargin 1005
latex-lab-block 35	\Roman 40, 45, 46
ltxcmd 35	\Roman 608
ltsockets 106	\roman 40, 45, 46
lua-visual-debug 54	\roman 609, 733, 5169
multicol 27, 135	S
scontents 27, 83, 84	\s 2926
shortlst 113, 118, 122	save-ans 2068
tagpdf 106	save-key 2366
\par 1280, 1308, 1346, 1518, 1622, 1628, 1667, 1672, 1694, 1699, 2653, 3731, 3892, 3910, 4170, 4173, 4313, 4527, 4542, 4588, 4602, 4845, 5100	save-ref 2288
para commands:	save-sep 2288
\para_end: 4862, 5118	scan commands:
\parbox 2295	\scan_stop: 4134, 4622, 4884, 5135, 5138
\parindent 4827, 5078	scontents internal commands:
\parsep 53, 109	\l_scontents_fname_out_tl 2885
\parsep 989, 3607, 4103, 4112	__scontents_parse_environment_keys:n 2891
parsep 929	__scontents_rescan_tokens:n 2898
\parskip 4828, 5079	\l_scontents_storing_bool 2883
\partopsep 3608, 3908, 4122	\l_scontents_writing_bool 2884
partopsep 929	seq commands:
peek commands:	\seq_clear:N 5288, 5431
\peek_meaning:NTF 4713, 4727, 4742, 4753, 4956, 4970, 4985	\seq_const_from_clist:Nn 5281
\peek_meaning_remove:NTF 4720, 4963	\seq_count:N 363, 4319, 5292
\peek_remove_spaces:n 3460	\seq_gclear:N 465, 466, 495, 496
	\seq_gput_right:Nn 451, 452, 481, 482, 2434
	\seq_if_empty:NTF 458, 488, 5194, 5306
	\seq_if_exist:NTF 2118, 5192
	\seq_if_in:NnTF 5198
	\seq_item:Nn 2923, 4306

\seq_map_function:NN	5297	stop-start-tags	3948, 3998
\seq_map_inline:Nn	5207, 5215, 5307, 5308	str commands:	
\seq_map_pairwise_function:NNN	460, 490	\c_backslash_str	2771, 5484, 5489, 5494, 5499, 5501, 5503, 5508, 5510, 5608, 5612, 5616, 5626, 5630, 5638, 5639, 5643, 5655, 5656, 5660, 5661, 5682, 5684, 5688, 5690, 5718, 5781, 5783, 5787, 5789, 5798, 5799, 5803, 5808, 5809, 5813, 5817, 5821
\seq_new:N	120, 121, 123, 138, 169, 170, 171, 172, 2121	\c_colon_str	2577, 3091, 5135
\seq_pop_left:NN	5296	\c_left_brace_str	5589, 5596, 5602
\seq_put_right:Nn	4198, 5304, 5320, 5441	\c_right_brace_str	5589, 5596, 5602
\seq_set_from_clist:Nn	5289	\str_case:nn	247, 306
\seq_set_map_e:NNn	5298	\str_case:nnTF	1848, 1856, 2405, 2413, 5235, 5244
\seq_use:Nn	199, 200, 5437	\str_clear:N	3641, 4654
series	1825	\str_count:n	234
\setcounter	891, 895, 897, 3565, 3610, 4167	\str_if_empty:NTF	1865, 1906, 1933
\setenumext	6, 131, 5286	\str_if_eq:nnTF	3566, 3612, 5337
\setenumextmeta	6, 132, 5327	\str_if_in:nnTF	5131
show-ans	2288, 2330	\str_new:N	80, 128, 143, 186
show-length	1102	\str_set:Nn	673, 679, 685, 704, 705, 706, 2308, 2309, 2335, 2336, 4055, 4058
show-pos	2330	\str_use:N	3337
skip commands:		\strut	3333, 3504
\skip_add:Nn	1240, 1249, 1258, 1271, 1275, 1299, 1303, 1319, 1377, 1379, 1393, 1396, 1417, 1419, 1433, 1436, 1456, 1458, 1472, 1475, 1494, 1543, 1544, 1555, 1557, 4112, 4120	\strutbox	1352, 1355, 1366, 1367, 1378, 1380, 1395, 1398, 1406, 1407, 1418, 1420, 1435, 1438, 1445, 1446, 1457, 1459, 1474, 1477, 1523, 1526, 1534, 1535, 1543, 1544, 1556, 1558, 1569, 1570, 1573, 1580, 1593, 1601, 1607, 1615, 4115, 4120, 4170, 4178, 4261
\skip_gset:Nn	1570, 1574, 1578		
\skip_gzero_new:N	1565, 1566		
\skip_horizontal:N	1073, 1085, 1097, 4799, 4811, 4849, 5062, 5104		
\skip_horizontal:n	1059, 2502, 2510, 3281, 3283, 4698, 4797, 4831, 4941, 5082		
\skip_if_eq:nnTF	1238, 1247, 1256, 1363, 1403, 1443, 1531, 1567, 1589, 1733, 1747, 1761, 1772, 1783, 1794, 1805, 1816		
\skip_new:N	77, 78, 79, 84, 85, 86, 87, 88, 89, 144, 191		
\skip_set:Nn	1223, 1227, 1285, 1289, 1313, 1366, 1367, 1385, 1406, 1407, 1425, 1445, 1446, 1464, 1488, 1534, 1535, 1549, 1569, 1573, 1591, 1595, 1599, 1605, 1609, 1613, 4096		
\skip_set_eq:NN	1324, 1325, 1327, 1334, 1499, 1500, 1501, 1506, 3563, 3606, 3607, 4828, 5079		
\skip_sub:Nn	1373, 1375, 1389, 1391, 1413, 1415, 1429, 1431, 1452, 1454, 1468, 1470, 1541, 1542, 1553, 1554		
\skip_use:N	1225, 1229, 1273, 1277, 1281, 1301, 1305, 1315, 1321, 1734, 1738, 1741, 1748, 1752, 1755, 3731		
\skip_vertical:N	573, 576, 997, 4526, 4540, 4864, 5120		
\skip_vertical:n	996, 4863, 5119		
\skip_zero:N	1333, 1347, 1485, 1486, 1487, 1505, 1519, 3608, 3714, 3877, 4122, 4123		
\skip_zero_new:N	1564, 1586, 1587, 1588		
\c_zero_skip	573, 576, 997, 1238, 1247, 1256, 1404, 1443, 1567, 1589, 1734, 1748, 1761, 1772, 1783, 1794, 1805, 1816, 4526, 4540, 4864, 5120		
\small	5157, 5161, 5165, 5169, 5173, 5177		
\smash	3333, 3504		
socket commands:			
\socket_assign_plug:nn	3978, 3986, 3994, 4030, 4038, 4046		
\socket_new:nn	3948, 3998		
\socket_new_plug:nnn	3949, 3957, 3965, 3999, 4007, 4016		
\socket_use:n	4031, 4039, 4047		
\socket_use:nn	3979, 3987, 3995		
start	903		
start*	903		
start-list-tags	3948, 3998		
\stepcounter	444, 474, 4106, 4248		
stop-list-tags	3948, 3998		
		T	
		tag commands:	
		\tag_mc_begin:n	3955, 4005, 4014
		\tag_mc_begin_pop:n	3971, 4023, 4162, 4164
		\tag_mc_end:	3959, 4009, 4018
		\tag_mc_end_push:	3952, 4002, 4150
		\tag_resume:n	3951, 4001, 4141, 4149, 4218, 4310, 4510, 4574
		\tag_struct_begin:n	3953, 3954, 3961, 3962, 3963, 4003, 4004, 4011, 4012, 4013, 4151
		\tag_struct_end:n	3960, 3967, 3968, 3969, 3970, 4010, 4019, 4020, 4021, 4022, 4161, 4163, 4628, 4890
		\tag_suspend:n	3972, 4024, 4132, 4143, 4156, 4209, 4302, 4620, 4882
		\tag_tool:n	4142
		T_EX and L^AT_EX 2_ε commands:	
		\@auxout	421
		\@currenvr	247, 306
		\protected@write	421
		tex commands:	
		\tex_newlinechar:D	2897
		text commands:	
		\text_expand:n	5127
		\textasteriskcentered	2305, 3357
		\textreferencemark	2322
		\thepage	427
		tl commands:	
		\c_space_tl	3173, 5533, 5548, 5571, 5575, 5762, 5763, 5772, 5773, 5833, 5837
		\tl_clear:N	671, 678, 2286, 2352, 2362, 2383, 2391, 2597, 2917, 2918, 3032, 3108, 4991
		\tl_clear_new:N	618
		\tl_const:Nn	46, 602
		\tl_gclear:N	354, 355, 356, 1886, 1891, 3007, 3326, 3346, 4546, 4606, 4800
		\tl_gclear_new:N	1873
		\tl_gput_right:Nn	603
		\tl_greplace_all:Nnn	624

<code>\tl_gset:Nn</code>	282, 283, 297, 298, 1874, 1887, 1892, 2111, 2921, 3257, 4748
<code>\tl_gset_eq:NN</code>	620, 3253, 4793
<code>\tl_if_blank:nTF</code>	2710, 2728, 2865, 3381, 3399, 3421, 4791, 5401
<code>\tl_if_empty:NTF</code>	738, 756, 784, 800, 819, 826, 852, 868, 1899, 1904, 1926, 1931, 1989, 2053, 2061, 2090, 2150, 2441, 2472, 2617, 2962, 2984, 3014, 3042, 3118, 3167, 3278, 4317, 4994, 5318
<code>\tl_if_empty:nTF</code>	1954
<code>\tl_if_exist:NTF</code>	1959
<code>\tl_if_novalue:nTF</code>	442, 472, 2724, 3040, 3116, 3152, 3232, 3251, 3259, 3431, 3639, 4087, 4652, 4922, 4992
<code>\tl_map_inline:Nn</code>	225, 621
<code>\tl_new:N</code>	38, 39, 40, 43, 48, 49, 52, 53, 59, 61, 62, 64, 65, 101, 102, 103, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 122, 124, 125, 126, 129, 132, 133, 151, 159, 160, 161, 164, 185
<code>\tl_put_left:Ne</code>	2951
<code>\tl_put_left:Nn</code>	2449, 2480, 2602, 2945, 2958, 2964, 2974, 3184, 3224, 4530, 4591, 5013, 5016
<code>\tl_put_right:Nn</code>	619, 748, 792, 808, 860, 2453, 2484, 2531, 2541, 2554, 2569, 2575, 2580, 2604, 2609, 2616, 2619, 2629, 2634, 2637, 2643, 3035, 3038, 3044, 3049, 3076, 3081, 3086, 3089, 3098, 3111, 3114, 3120, 3125, 3135, 4996, 5000
<code>\tl_remove_all:Nn</code>	5317
<code>\tl_remove_once:Nn</code>	2519, 3061
<code>\tl_replace_all:Nnn</code>	623, 5352
<code>\tl_reverse:N</code>	2518, 2520, 3060, 3062
<code>\tl_set:Nn</code>	54, 251, 261, 310, 311, 318, 319, 326, 327, 588, 672, 677, 683, 684, 737, 781, 851, 1056, 1070, 1083, 1095, 1988, 2089, 2353, 2363, 2384, 2392, 2674, 2885, 3154, 3199, 3212, 5002, 5025, 5315, 5351, 5421
<code>\tl_set_eq:NN</code>	629, 743, 746, 789, 791, 805, 807, 857, 859, 2517, 3059, 3072, 3443, 3448, 4236, 4238
<code>\tl_to_str:n</code>	1959, 1965, 1970, 5127
<code>\tl_trim_spaces:n</code>	619, 5304, 5315, 5321, 5337
<code>\tl_use:N</code>	625, 628, 758, 821, 828, 870, 1128, 1132, 1136, 1140, 1144, 1148, 1152, 1156, 1160, 1164, 1168, 1172, 1176, 1180, 1184, 1188, 2507, 2524, 2532, 2543, 2556, 2561, 2572, 3240, 3246, 3274, 3317, 3318, 3325, 3339, 3434, 3438, 3446, 3490, 3491, 3497, 3508, 3803, 3936, 4241, 4537, 4598, 4804, 4832, 4833, 5055, 5083, 5088, 5182, 5183, 5184, 5185, 5186, 5203, 5300, 5419
token commands:	
<code>\token_to_str:N</code>	423
<code>\topsep</code>	3908, 4120
<code>topsep</code>	929
<code>\topskip</code>	1333, 1505
U	
<code>\u</code>	228, 2926
<code>\unkern</code>	242
<code>unknown</code>	3367, 3389, 3407
<code>\unskip</code>	241
use commands:	
<code>\use:N</code>	235, 3322, 3343, 3805
<code>\use:n</code>	1839, 2396, 5133, 5226
<code>\use_none:nn</code>	415, 5358
<code>\usecounter</code>	3564, 3609
V	
<code>\value</code>	1902, 1908, 1915, 1921, 1929, 1935, 1942, 1948
vbox commands:	
<code>\vbox_set:Nn</code>	4211
<code>\vbox_set_top:Nn</code>	4535, 4596
<code>\vspace</code>	989, 1738, 1741, 1752, 1755, 1765, 1767, 1776, 1778, 1787, 1789, 1798, 1800, 1809, 1811, 1820, 1822
W	
<code>widest</code>	903
<code>wrap-ans</code>	2288
<code>wrap-label</code>	642
<code>wrap-label*</code>	642
<code>wrap-opt</code>	2288
Z	
<code>\z</code>	2926