enumext

ENUMERATE EXERCISE SHEETS

V1.9   2025-11-01

©2024–2025 by Pablo González L†

Abstract

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

Contents

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 are notes on the main ones:

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

---

License and Requirements

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

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

💣 The minimum requirement is LaTeX release 2025-11-01.

1 Introduction

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

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

1. Factor $x^2 - 2x + 1$
2. Factor $3x + 3y + 3z$
3. True False
   (a) $\alpha > \delta$
   (b) LaTeX2e is cool?
4. Related to Linux

(a) You use Linux?
(b) Usually uses the package manager?
(c) Rate the following package and class
   i. `xsim-exam`
   ii. `xsim`
   iii. `exsheets`

Sometimes we are also interested in showing the *"answers"* along with the questions:

1. Factor $x^2 - 2x + 1$
   * $(x-1)^2$
2. Factor $3x + 3y + 3z$
   * $3(x+y+z)$
3. True False
   (a) $\alpha > \delta$
      * False
   (b) LaTeX2e is cool?
      * Very True!
4. Related to Linux

(a) You use Linux?
   * Yes
(b) Usually uses the package manager?
   * Yes, `dnf`
(c) Rate the following package and class
   i. `xsim-exam`
      * doesn't exist for now :(
   ii. `xsim`
      * very good
   iii. `exsheets`
      * obsolete

Or we are interested in referring to a specific question and its *"answer"*, for example:

The answer to 3.(b) is "Very True!" and the answer to 4.(c).ii is "very good".

Or we are interested in printing all the *"answers"*:

1. $(x-1)^2$ ※
2. $3(x+y+z)$ ※
3. (a) False ※
   (b) Very True! ※
4. (a) Yes ※

(b) Yes, `dnf` ※
(c) i. doesn't exist for now :(※
   ii. very good ※
   iii. obsolete ※

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

1. First type of questions
   A) value   C) value
   B) correct   D) value

2. Second type of questions
   I. $2\alpha + 2\delta = 90°$
   II. $\alpha = \delta$
   III. $\angle EDF = 45°$

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

★ 3. Third type of questions
   (1) $2\alpha + 2\delta = 90°$
   (2) $\angle EDF = 45°$

   A) value   D) value
   B) value   E) value
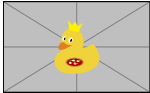   C) value

4. Question with image and label below:

A)   B)   C)

D)   E)

5. Question with image on right side:

   A) value
   B) value
   C) value
   D) correct
   E) value

Where we are interested in the ⟨*label*⟩ and a *"short note"* that we leave as an explanation, and then print them:

1. B) $x = 5$
2. D)
3. C) some note
4. E) A duck
5. D) "other note"

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

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

1.1 Description and usage

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

⬤ This package is NOT intended 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 MiKTEX, 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"*, which makes it difficult to obtain the desired *horizontal spaces* in a list environment. Usually we don't want the list to go beyond the left margin of the page, but since these four values are related, that causes a problem.

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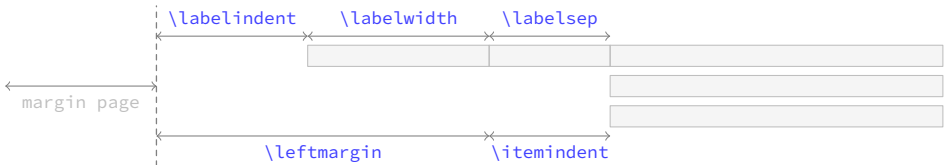
Figure 1: Representation of horizontal lengths in enumitem.

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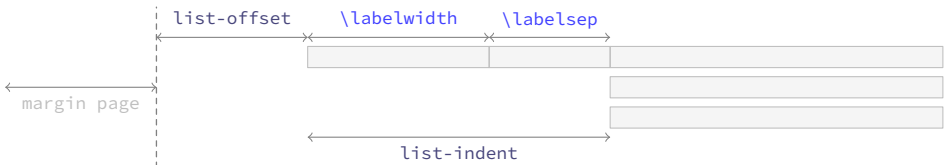
Figure 2: Representation of horizontal lengths concept in enumext.

In this way we reduce a *little* the amount of parameters we have to pass. With the default values of keys list-offset, list-indent, labelwidth and labelsep the lists will have the (usually) expected output for *"simple worksheets"*. The figure 3 shows the visual representation.
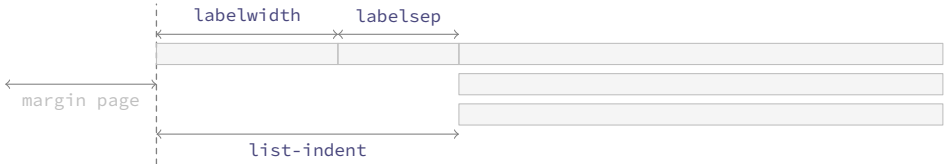
Figure 3: Default horizontal lengths list-offset=0pt, list-indent=\labelwidth+\labelsep in enumext.

1.3 User interface

The user interface consists of two main list environments `enumext` (vertical) and `enumext*` (horizontal), the environment `anskey*` and the command `\anskey` to *"store content"* and the environments `keyans`, `keyans*` and `keyanspic` for multiple choice. It also provides the commands `\getkeyans` to print individual *stored content*, `\printkeyans` to print all *stored content*, `\foreachkeyans` to print a range of *stored content*, `\miniright` for `minipage`, `\setenumext` to config [⟨*key* = *val*⟩], `\setenumextmeta` to add a *"meta-key"* and `\resetenumext` to reset counters.

1.3.1 Public counters

The package `enumext` uses 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 a *rigid length* and contains the *"width of the content"* of each `\item` regardless of `labelwidth` and `labelsep`.

🎯 If any package defines `\itemwidth` or the user defines `\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 for a two-column output as shown in the figure 4.
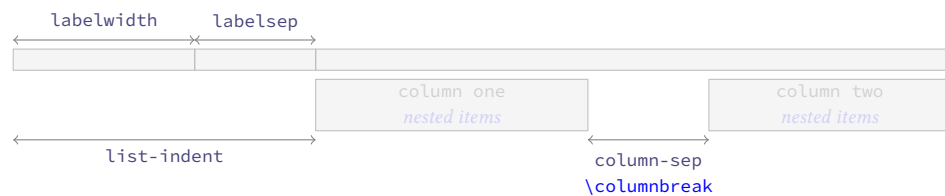
Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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 §5.7).

🎯 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 the *"left side"* and *"right side"* are 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 §5.8).

1.3.5 The `\label` and `\ref` system

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

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, it is only used for typesetting. The implementation supports `\footnote` and is compatible with `hyperref` and works the same way as if used anywhere in the document.

Unfortunately, if `hyperref` is loaded and *tagged* PDF is not active, it will not produce the "links", the internal implementation uses `\footnotetext[`⟨*number*⟩`]` and `\footnotemark[`⟨*number*⟩`]{`⟨*text*⟩`}` and "links" for this are not supported by the `hyperref` package.

The best way to solve this if *tagged* PDF is NOT active is to use Jean-François Burnol's `footnotehyper`[9] package. It will support keeping the "links" if `hyperref` is loaded with the `hyperfootnotes=true` option (default). Load it is as follows:

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

🏷 At the moment the `footnotehyper` package v1.1e (2021/08/13) is not compatible with *tagged* PDF.

2   The environments provided

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

`enumext`
`enumext*`

```
\begin{enumext}[⟨keyval list⟩]              \begin{enumext*}[⟨keyval list⟩]
  \item ⟨item content⟩                        \item ⟨item content⟩
  \item [⟨custom⟩] ⟨item content⟩             \item [⟨custom⟩] ⟨item content⟩
  \item*[⟨symbol⟩][⟨offset⟩]  ⟨item content⟩  \item*[⟨symbol⟩][⟨offset⟩] ⟨item content⟩
\end{enumext}                               \end{enumext*}
```

2.1   The environment `enumext`

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

Example with `columns=2`

1. This text is in the first level.

  (a) This text is in the second level.

      i.   This text is in the third level.

A. This text is in the fourth level.

X This text is in the first level.

⋆ 2. This text is in the first level.

2.2   The environment `enumext*`

The `enumext*` is a *horizontal list environment* similar to the `shortenumerate` or `tasks` environments provided by the `shortlst`[16] and `tasks`[17] packages, `\item` and `\item[`⟨*custom*⟩`]` work as usual. The options can be configured globally using `\setenumext` command and locally using `[`⟨*key = val*⟩`]` in the environment.

Some considerations to take into account for this environment:

- The environment cannot be nested within itself or in the environment `keyans*`, but it can be nested within `enumext` and vice versa.
- Each *"item content"* in the environment is placed within a `minipage` environment whose *width* is stored in the dimension `\itemwidth` does NOT include `labelwidth` and `labelsep`, only the *width of the content*.
- You cannot have floating environments like `figure` or `table` or `\marginpar` but `\footnote` with `hyperref` is supported. If you want to activate links *"without"* *tagged* PDF active you must load `footnotehyper` and `hyperref` packages (see §1.3.6).
- 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 places 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*. It can be configured with the keys `item-sym*` and `item-pos*` locally in the environment or globally using `\setenumext` command (§3).

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

2.3.1 Keys for `\item*`

item-sym* = {⟨symbol⟩}   default: *\textborn*

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

item-pos* = {⟨rigid length⟩}   default: *by levels*

Sets the *offset* between the box containing the current ⟨*label*⟩ defined by `labelwidth` key and the ⟨*symbol*⟩ set by `item-sym*` key. The default values are set by `labelsep` key at each level. If positive values are passed, it will *offset to the left*; 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 a *"first optional argument"* `\item(`⟨*columns*⟩`)` which *"joins items"* between columns. Let's consider the following examples adapted directly from the `tasks` 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 9. The ninth
two columns (196.17749pt)

Z The tenth (89.28171pt)

3 The command `\setenumext`

\setenumext

`\setenumext{`⟨*key = val*⟩`}`	`\setenumext[`⟨*keyans**⟩`]{`⟨*key = val*⟩`}`
`\setenumext[`⟨*enumext, level*⟩`]{`⟨*key = val*⟩`}`	`\setenumext[`⟨*print, level*⟩`]{`⟨*key = val*⟩`}`
`\setenumext[`⟨*enumext**⟩`]{`⟨*key = val*⟩`}`	`\setenumext[`⟨*print, **⟩`]{`⟨*key = val*⟩`}`
`\setenumext[`⟨*keyans*⟩`]{`⟨*key = val*⟩`}`	`\setenumext[`⟨*print**⟩`]{`⟨*key = val*⟩`}`

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

The ⟨*keys*⟩ set in the *optional argument* of environments and commands have the *highest precedence*, overriding both options passed by `\setenumext`. If the *optional argument* is not passed, the *first level* of the environment `enumext` will be taken by default.

🎯 For security reasons the keys `resume` *"with value"*, `resume*`, `reset`, `reset*`, `series` and `save-ans` they can NOT be set by this command and are ignored. The key `save-ans` that activate the *"storage system"* must be passed directly in the *optional argument* of the *"first level"* of the environment in which they are executed.

4 The command `\setenumextmeta`

\setenumextmeta

```
\setenumextmeta[⟨1⟩]{⟨key name⟩}{⟨key-one = val, key-two = val, ...⟩}
\setenumextmeta[⟨2⟩]{⟨key name⟩}{⟨key-one = val, key-two = val, ...⟩}
\setenumextmeta[⟨3⟩]{⟨key name⟩}{⟨key-one = val, key-two = val, ...⟩}
\setenumextmeta[⟨4⟩]{⟨key name⟩}{⟨key-one = val, key-two = val, ...⟩}
\setenumextmeta[⟨*⟩]{⟨key name⟩}{⟨key-one = val, key-two = val, ...⟩}
\setenumextmeta*{⟨key name⟩}{⟨key-one = val, key-two = val, ...⟩}
```

The command `\setenumextmeta` adds a new *"meta-key"* for the environments `enumext` and `enumext*`, the {⟨*key name*⟩} must be different from those defined by the package. The *optional argument* of the form `[1]`, `[2]`, `[3]`, `[4]` adds a new *"meta-key"* for levels `1`, `2`, `3` and `4` of the `enumext` environment, the form `[*]` adds a new *"meta-key"* for the `enumext*` environment. If it is run *without* the *optional argument*, it will adds a new *"meta-key"* for the *"first level"* of the `enumext` environment.

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 ⟨*key = val*⟩ system used by the enum**ext** package is implemented using `l3keys` so it must be noted that ⟨*keys*⟩ marked as *"value forbidden"*, that is ⟨*key*⟩ is different from ⟨*key=*⟩.

All ⟨*keys*⟩ described in this section are available for the `enumext`, `enumext*`, `keyans` and `keyans*` environments with the exception of the keys `series`, `resume`, `resume*`, `reset` and `reset*` which are only available for the `enumext` and `enumext*` environments; and the keys `mini-right`, `mini-right*` which are only available for the `enumext*` and `keyans*` environments.

All ⟨*keys*⟩ 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 ⟨*key*⟩ that sets *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 ⟨*value forbidden*⟩   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` is redefined using `\makebox` by default. If `enumext` or `keyans` is used in the `enumext*` environment the key must be activated manually.

label = {⟨ `\alph*` | `\Alph*` | `\arabic*` | `\roman*` | `\Roman*` ⟩}   default: *by levels*

Sets the ⟨*label*⟩ that will be printed at the *current level* and default value for `labelwidth` key. The default value for the first level of the environments `enumext` and `enumext*` is `\arabic*.`, for second level are (`\alph*`), for third level are `\roman*.` and for fourth level is `\Alph*.`. For `keyans` and `keyans*` environments the default value is `\Alph*)`.

🎯 This key is intended to give the basic structure with which the ⟨*label*⟩ 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 ⟨*label*⟩ as an argument, for example `\emph{`⟨`\alph*`⟩`}` will return an error. For full customization of how ⟨*label*⟩ is displayed use the `font`, `wrap-label` and/or `wrap-label*` keys.

labelsep = {⟨*rigid length*⟩}   default: `0.3333em`

Sets the *horizontal space* between the box containing the current ⟨*label*⟩ 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 = {⟨*rigid length*⟩}   default: *by label*

Sets the *width* of the label box containing the current ⟨*label*⟩ set by the `label` key. Internally sets the value of `\labelwidth` for the current level. The default values are calculated by means of the *width* of a box by setting a *value* to the current counter set by `label` key using '`0`' for `\arabic*`, '`M`' for `\Alph*`, '`m`' for `\alph*`, '`VIII`' for `\Roman*` and '`viii`' for `\roman*`.

widest = {⟨*integer* | *string*⟩}   default: *empty*

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

font = {⟨*font commands*⟩}   default: *empty*

Sets the *font style* for the current ⟨*label*⟩ defined by `label` key. You cannot use commands with an argument, for example `font={\small\textbf{#1}}` will return an error, but `font={\bfseries\small}` it will work. For full customization of how ⟨*label*⟩ is displayed use `wrap-label` and/or `wrap-label*` keys.

align = {⟨*left* | *right* | *center*⟩}   default: *left*

Sets the *aligned* of ⟨*label*⟩ defined by `label` key on the current level in the label box of width set by `labelwidth`.

wrap-label = {⟨*code* `{#1}` *more code*⟩}   default: *empty*

Wraps the *current* ⟨*label*⟩ defined by `label` key referenced by `{#1}` after executing the `align` and `font` keys. The `{`⟨*code*⟩`}` must be passed between braces and it does not modify the value set by the `labelwidth` key and is applied *only* on `\item` and `\item*`. When using it in the `\setenumext` command it is necessary to use the *double* '`{##1}`'. For example `wrap-label={\fbox{#1}}` or you can create a command:

```
\NewDocumentCommand \mywrap { s m }
  {
    \IfBooleanTF{#1}
      {\textcolor{red}{\textbf{Q}}\textcolor{blue}{\textbf{.}}\textcolor{gray}{#2}}
      {\textcolor{blue}{\textbf{Q}}\textcolor{red}{\textbf{.}}\textcolor{gray}{#2}}
  }
```

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

wrap-label* = {⟨*code* {#1} *more code*⟩}   default: *empty*

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

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

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

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

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

5.2 Keys for penalties

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

beginpenalty = {⟨*integer*⟩}   default: *-51*

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

midpenalty = {⟨*integer*⟩}   default: *-51*

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

endpenalty = {⟨*integer*⟩}   default: *-51*

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

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

5.3 Keys for spaces

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

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

5.3.1 Vertical spaces

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

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

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

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

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

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

Set the *vertical space* added, beyond `topsep`, to the "top" and "bottom" of the entire environment if the environment instance is preceded by a "*blank line*" or `\par` command. Internally sets the value of `\partopsep` for the current level. The default values for first and second level in environment `enumext` are `2.0pt plus 1.0pt minus 1.0pt`, for third and fourth level are `1.0pt minus 1.0pt`. For the `keyans` environment the

default value is `2.0pt plus 1.0pt minus 1.0pt`, and for the `keyans*` and `enumext*` environments it is available but *without* effect.

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

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

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

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

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

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

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

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

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

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

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

Extra vertical spaces

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

`above` = {⟨*rubber length* | *rigid length*⟩} default: *not used*

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

`above*` = {⟨*rubber length* | *rigid length*⟩} default: *not used*

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

`below` = {⟨*rubber length* | *rigid length*⟩} default: *not used*

Set the *extra vertical space* 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*` = {⟨*rubber length* | *rigid length*⟩} 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"*.

5.3.2 Horizontal spaces

`list-offset` = {⟨*rigid length*⟩} 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` = {⟨*rigid length*⟩} 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 `labelwidth` plus `labelsep`. Passing a value to this key is equivalent to setting the value for the `list-offset` key.

itemindent = {⟨*rigid length*⟩} default: 0pt

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

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

rightmargin = {⟨*rigid length*⟩} default: 0pt

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

listparindent = {⟨*rigid length*⟩} default: 0pt

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

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

5.4 Keys for add code

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

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

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

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

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

first = {⟨*code*⟩} default: *not used*

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

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

after = {⟨*code*⟩} default: *not used*

Execute {⟨*code*⟩} *"after"* finishing the environment. The {⟨*code*⟩} must be passed between braces.

5.5 Keys for start, series and resume

start = {⟨*integer | integer expression*⟩} default: 1

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

start* = {⟨*integer | string*⟩} default: *not used*

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

🖐 For compatibility with *tagged* PDF, the *start value* are set *"after"* the *second argument* to the `list` environment and *"before"* the execution of the first `\item` and the `first` key: `\begin{list}`{⟨*arg one*⟩}{⟨*arg two*⟩}`\setcounter{enumX}\item`.

🖐 The following ⟨*keys*⟩ are available only for the `enumext` and `enumext*` environments.

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

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

🖐 For security reasons the `series` key will never save in {⟨*series name*⟩} the ⟨*keys*⟩ `series`, `resume`, `resume*`, `reset`, `reset*`, `save-ans`, `save-key`, `start*` and `start`.

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

Sets the *start value* and *options* for the *"current level"* continuing the numbering and *options* of the *"same level"* as the environment in which the `series={`⟨*series name*⟩`}` key was executed, the *start value* will continue numbering according to the last execution of `resume={`⟨*series name*⟩`}`. If passed *"without value"* this will only set *start value* continue the numbering of the *"same level"* from the last environment and level in which `series={`⟨*series name*⟩`}` or `resume={`⟨*series name*⟩`}` is NOT present and if the `save-ans` key is active (on the left) 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.

🎯 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*.

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

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

🎯 When using the key `resume={`⟨*series name*⟩`}` or `resume*` you will have hierarchy in the ⟨*keys*⟩ that are stored in {⟨*series name*⟩} or in an internal version of {⟨*series name*⟩} in the case of `resume*`. If you want to *reset* the value of a ⟨*key*⟩ that is already stored in {⟨*series name*⟩} or in an internal version of {⟨*series name*⟩} this must be placed to the *right* of the key `resume={`⟨*series name*⟩`}` or `resume*`.

🎯 When the `resume*` key is executed consecutively, it does not rewrite the ⟨*keys*⟩ stored in the internal version of {⟨*series name*⟩} and if the environment that precedes it does not have the *optional argument*, it will just continue with the numbering.

5.6 Keys for `reset`

reset ⟨*value forbidden*⟩   default: *not used*

Resets the *start value* of the *"counters"* in the `enumext` and `enumext*` environments along with the *"internal counters"* used by the `resume` *without value* and `resume*` keys at the *"level"* at which it is executed. The *start value* can be overwritten using the `start` or `start*` keys.

reset* ⟨*value forbidden*⟩   default: *not used*

Resets the *start value* of the *"counters"* in the `enumext` and `enumext*` environments along with the *"internal counters"* used by the `resume` *without value* and `resume*` keys at the *"level"* at which it is executed and in the *"levels below"* it in the case of the `enumext` environment. The *start value* can be overwritten using the `start` or `start*` keys.

🎯 These keys are intended to be used in cases where the `\resetenumext` command does not work, e.g. after an unnumbered chapter. It should preferably be set *only* on the *first level*, although it is available for all levels.

5.6.1 The command `\resetenumext`

`\resetenumext`

`\resetenumext[`⟨1⟩`]{`⟨*some counter*⟩`}`   `\resetenumext[`⟨4⟩`]{`⟨*some counter*⟩`}`
`\resetenumext[`⟨2⟩`]{`⟨*some counter*⟩`}`   `\resetenumext[`⟨*⟩`]{`⟨*some counter*⟩`}`
`\resetenumext[`⟨3⟩`]{`⟨*some counter*⟩`}`   `\resetenumext*{`⟨*some counter*⟩`}`

The `\resetenumext` command *"resets"* the *start value* of the *"counters"* for the `enumext` and `enumext*` environments along with the *"internal counters"* used by the keys `resume` *without value* and `resume*` according to the value of {⟨*some counter*⟩}. For example `\resetenumext{chapter}` will *"reset"* the numbering of *"all levels"* of the `enumext` environment for each execution of a *"numbered"* chapter.

The *optional argument* of the form `[1]`, `[2]`, `[3]`, `[4]` *"reset"* the values for levels 1, 2, 3 and 4 of the `enumext` environment, the form `[*]` *"reset"* the values for the `enumext*` environment. If is run *without* the *optional argument*, it will *"reset"* the values for *"all levels"* of the `enumext` environment.

The *starred argument* '`*`' will *"reset"* the values for *"all levels"* of the `enumext` and `enumext*` environments.

5.7 Keys for `multicols`

columns = {⟨*integer*⟩}   default: *1*

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

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

Set the *space between* columns used by the `multicols` environment within the environments `enumext` and `keyans`. Internally sets the value of `\columnsep`, by default its value is equal to the sum of the values set in the keys `labelwidth` and `labelsep` of the current level. In the `enumext*` and `keyans*` environments they correspond to the *space between* columns (without joining) and internally adjust the value of `\itemwidth`.

5.8 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`.

5.8.1 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 LATEX justification is maintained in the `minipage` on the *"right side"*.

5.8.2 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⟩}]
   \item Text \anskey{answer}
   \item Text
     \begin{keyans}
        ...
     \end{keyans}
\end{enumext}
```

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

By executing the key `save-ans={`⟨*store name*⟩`}` the entire *"structure"* of the environment (excluding the *first level*) including the *optional 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* {⟨*store name*⟩} set by `save-ans` key. The ⟨*key list*⟩ passed to this key ignores any ⟨*keys*⟩ in the *"stored structure"* and must be passed between braces. For example, if we execute at a second level:

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

The *"stored keys"* by default in the *sequence* {⟨*store name*⟩} would be `nosep` and `columns=2`, but using the key `save-key={columns=3}` will overwrite and the *"stored key"* in the *sequence* {⟨*store name*⟩} are only `columns=3` ignoring all the others.

save-sep = {⟨*text symbol*⟩}   default: *{, }*

Sets the *text symbol* that will separate the current ⟨*label*⟩ 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* {⟨*store name*⟩} set by `save-ans` key. The {⟨*text symbol*⟩} must always be passed between braces, whitespace '␣' is preserved within the braces and only affects the *"stored content"* and not what is displayed when using the `show-ans` or `show-pos` keys.

no-store   ⟨*value forbidden*⟩   default: *not used*

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

6.1.1   Keys for `label` and `ref`

save-ref = {⟨*true* | *false*⟩}   default: *false*

Activates the *"internal label and ref"* mechanism for referencing *"stored content"* in *prop list* {⟨*store name*⟩} set by `save-ans` key. To reference the location of the *"stored content"* within the environment you must use `\ref{`⟨*store name* : *position*⟩`}`, where ⟨*position*⟩ corresponds to the position occupied by the *"stored content"* in the *prop list* {⟨*store name*⟩} 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 = {⟨*symbol*⟩}   default: *\textreferencemark*

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

6.1.2   Keys for `wrap` and `marks`

The enum**ext** package provides a set of ⟨*keys*⟩ to set and manipulate *"symbol marks"* associated with *"answers"* and how they are displayed and stored in the *sequence* and *prop list*.

The ⟨*keys*⟩ available for the `\anskey` command and the `anskey*` environment can be passed *"only"* in the *optional argument* in the *"first level"* of the `enumext` or `enumext*` environment.

The ⟨*keys*⟩ available for the `keyans` and `keyans*` environments can be passed locally in the *optional argument*, at the *"first level"* of the `enumext` or `enumext*` environment or via the `\setenumext` command with one minor difference, when ⟨*keys*⟩ are passed through the *"first level"* of the `enumext` or `enumext*` environment they are set in *"both"* environments, but when they are passed using the `\setenumext` command they are set *"individually"* in each environment.

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

Display the *symbol* set by the `mark-ans` key to the left of the *mandatory argument* ⟨*content*⟩ passed to the `\anskey` command and ⟨*body*⟩ for the `anskey*` environment using the `wrap-ans` key if set. For `\item*` and `\anspic*` the `keyans`, `keyans*` and `keyanspic` environments it will display the *symbol* set by the `mark-ans*` key to the left of the current ⟨*label*⟩ and *optional argument*. If the *optional argument* is present in `\item*` or `\anspic*` it will be shown using `wrap-opt` key.

Keys for `\anskey` and `anskey*`

mark-ans = {⟨*symbol*⟩}   default: *\textasteriskcentered*

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

mark-pos = {⟨*left* | *right* | *center*⟩}   default: *left*

Sets the *aligned* of the *"symbol"* defined by `mark-ans` key for `\anskey` command and `anskey*` environment. The *"symbol"* is aligned in a box with the same dimensions of the label box defined by `labelwidth` key on the current level and separated by the value of the `mark-sep` key.

mark-sep = {⟨*rigid length*⟩}   default: *labelsep*

Sets the *horizontal space* between the box containing the *"symbol"* defined by `mark-ans` key and the *mandatory argument* ⟨*content*⟩ passed to the `\anskey` command and the ⟨*body*⟩ in `anskey*` environment.

wrap-ans = {⟨*code* {#1} *more code*⟩}   default: *\fbox+\parbox{#1}*

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

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

mark-ans* = {⟨*symbol*⟩}   default: *\textasteriskcentered*

Sets the *symbol* to be displayed in the left margin for `\item*` and `\anspic*` for the `keyans`, `keyans*` and `keyanspic` environments when using the key `show-ans`. The *"symbol"* is placed in a box of width equal to the value of `labelwidth` of the environment in which it is executed, separated by the value of the key `mark-sep*` and aligned by the value of the key `mark-pos*`. This key is not affected by the keys `font` or `wrap-label` so if you want to apply *styling* you have to do it directly, for example: `mark-ans*={\textcolor`{red}`{\textbf`{`\textasteriskcentered`}}}.

mark-pos* = {⟨*left* | *right* | *center*⟩}   default: *left*

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

mark-sep* = {⟨*rigid length*⟩}   default: *labelsep*

Sets the *horizontal space* between the box containing the *"symbol"* defined by `mark-ans*` key and the current ⟨*label*⟩ for `\item*` and `\anspic*` in the `keyans`, `keyans*` and `keyanspic` environments.

wrap-ans* = {⟨*code* {#1} *more code*⟩}   default: *not used*

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

wrap-opt = {⟨*code* {#1} *more code*⟩}   default: *[[#1]]*

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

6.1.3   Keys for debug and checking

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

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

check-ans = {⟨*true* | *false*⟩}   default: *false*

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

6.2   The command `\anskey`

`\anskey`   `\anskey`[⟨*keys*⟩]{⟨*content*⟩}

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

If `save-ref` key are active and the `hyperref`[8] package is detected, `\hyperlink` and `\hypertarget` will be used, otherwise the usual *"label and ref"* system provided by LaTeX will be used.

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

6.2.1 Keys for `\anskey`

By default the *mandatory argument* ⟨*content*⟩ passed to `\anskey` when *"storing"* in the *sequence* {⟨*store name*⟩} has the form `\item` ⟨*content*⟩, the following ⟨*keys*⟩ allow modifying the way in which it is *"stored"* in the *sequence.*

`break-col` ⟨*value forbidden*⟩   default: *not used*

Stores {⟨*content*⟩} in the *sequence* {⟨*store name*⟩} of the form `\columnbreak \item` ⟨*content*⟩.

`item-join` = {⟨*columns*⟩}   default: *not set*

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

`item-star` ⟨*value forbidden*⟩   default: *not used*

Stores {⟨*content*⟩} in the *sequence* {⟨*store name*⟩} of the form `\item*` ⟨*content*⟩.

`item-sym*` = {⟨*symbol*⟩}   default: *not set*

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

`item-pos*` = {⟨*rigid length*⟩}   default: *not set*

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

Example

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

★ 1. Text containing our instructions or questions.
   ∗ first answer

2. Text containing our instructions or questions.
   (a) Question.
      ∗ second answer

3. Text containing our instructions or questions.
   ∗ third answer

4. Text containing our instructions or questions.
   ∗ fourth answer

6.3 The environment anskey*

`anskey*` `\begin{anskey*}`[⟨*key = val*⟩] ⟨*body content*⟩ `\end{anskey*}`

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

By design the environment cannot be nested but full supports *"verbatim material"* in the ⟨*body*⟩ and it is assumed that *"each numbered"* `\item` or `\item*` within the environment in which it is active it has a *"single execution"* unless `\item` or `\item*` open a nested level or use the `no-store` key.

The `anskey*` environment is implemented using the new *"collect code"* `c-type` argument part of LaTeX release 2025-06-01[13]. `\begin{anskey*}` and `\end{anskey*}` must be in different lines and should not appear within verbatim environments or commands. All ⟨*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, LaTeX will return an "error" message if this happens. In a similar way comments "`%`" or "any character" after `\end{anskey*}` on the same line LaTeX will return a "warning" message.

6.3.1 Keys for anskey*

The `anskey*` environment uses the same ⟨*keys*⟩ as the `\anskey` command next to the ⟨*keys*⟩ `write-env`, `overwrite` and `force-eol`. The environment is available for all levels of the `enumext` environment and the `enumext*` environment, but it is disabled for the `keyans`, `keyans*` and `keyanspic` environments.

`write-env` = {⟨*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 *last 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}`%.

Example

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

  \item Text containing our instructions or questions.
    \begin{enumext}
      \item Question.
        \begin{anskey*}
          ⟨second answer⟩
        \end{anskey*}
    \end{enumext}

  \item Text containing our instructions or questions.
    \begin{anskey*}
      ⟨third answer⟩
    \end{anskey*}

  \item Text containing our instructions or questions.
    \begin{anskey*}
      ⟨fourth answer⟩
    \end{anskey*}
\end{enumext}
```

⋆ 5. Text containing our instructions or questions.
[5] First answer with `verbatim`
6. Text containing our instructions or questions.
  (a) Question.
    [6] second answer

7. Text containing our instructions or questions.
[7] third answer
8. Text containing our instructions or questions.
[8] fourth answer

6.4   The environments keyans and keyans*

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

The `keyans` and `keyans*` environments are *"enumerated list"* environments designed for *"multiple choice"* questions activated by the `save-ans` key.

This environments can NOT be nested and must always be at the *"first level"* of the `enumext` environment, the commands `\item` and `\item[`⟨*custom*⟩`]` work in the usual and the command `\item(`⟨*columns*⟩`)` is available for the `keyans*` environment.

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

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

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

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

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

6.4.1 The \item* in keyans and keyans*

\item*
```
\item*
\item*[⟨content⟩]
```

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

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

Example

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

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

  \item Text containing a question and image.

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

1. Text containing a question.
   - A) Choice
   - C) Choice
   - E) Choice
   - ∗ B) Correct choice
   - D) Choice

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

   Some text

6.5 The environment keyanspic

keyanspic
```
\begin{keyanspic}[⟨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 ⟨*label*⟩ as the keyans environment that uses the \anspic command instead of \item. It is intended for placing *drawings or tabular* with ⟨*label*⟩ centered *above* or *below* in a *single line* or *upper and lower* layout style.

When the keyanspic environment is used *without keys* the ⟨*labels*⟩ are centered *below* the *drawings or tabular* in a *single line* layout style.

A representation of the output can be seen in the figure 6.
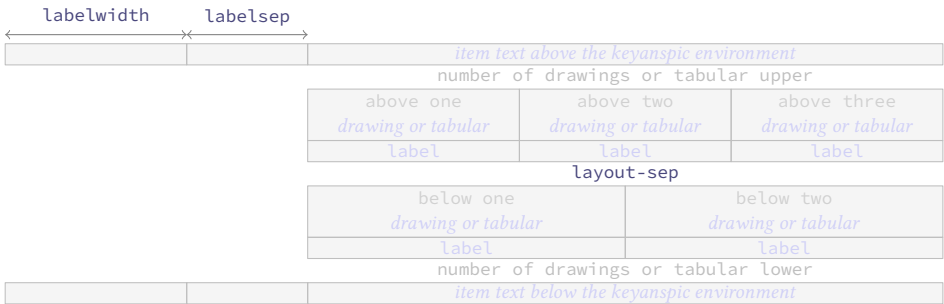
Figure 6: Representation of the keyanspic environment with layout-sty={⟨*3, 2*⟩} in enumext.

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.

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

6.5.2 The command \anspic

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

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

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

Example

```
\begin{enumext}[save-ans=test,show-ans=true,nosep]
  \item Question with images and labels below.

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

  \item Question with images and labels above.

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

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

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

\end{enumext}
```

1. Question with images and labels below.

A)

B)

C)
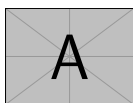
D)

∗ E) [note]

2. Question with images and labels above.
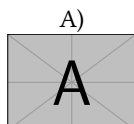
A)
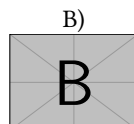
B)
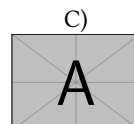
C)
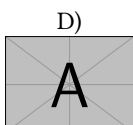
D)
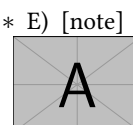
∗ E) [note]

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

A)

B)

C)

D)

∗ E) [note]

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

6.6 Printing stored content

6.6.1 The command `\getkeyans`

`\getkeyans`   `\getkeyans{⟨store name : position⟩}`

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

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

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

6.6.2 The command `\foreachkeyans`

`\foreachkeyans`   `\foreachkeyans[⟨key = val⟩]{⟨store name⟩}`

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

Options for command

`sep = {⟨code⟩}`   default: *{; }*

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

`step = {⟨integer⟩}`   default: *1*

Sets the *step* (increment) applied to the value set by key `start` for *"each"* `{⟨content⟩}` stored in *prop list* `{⟨store name⟩}`. The value must be a ⟨*positive integer*⟩.

`start = {⟨integer⟩}`   default: *1*

Sets the *position* of the *prop list* `{⟨store name⟩}` from which execution will start. The value must be a ⟨*positive integer*⟩.

`stop = {⟨integer⟩}`   default: *0*

Sets the *position* of the *prop list* `{⟨store name⟩}` from which execution will finish. The value must be a ⟨*positive integer*⟩.

before = {⟨*code*⟩}   default: *empty*

> Sets the {⟨*code*⟩} that will be executed ⟨*before*⟩ each {⟨*content*⟩} stored in *prop list* {⟨*store name*⟩}. The {⟨*code*⟩} must be passed between braces.

after = {⟨*code*⟩}   default: *empty*

> Sets the {⟨*code*⟩} that will be executed ⟨*after*⟩ each {⟨*content*⟩} stored in *prop list* {⟨*store name*⟩}. The {⟨*code*⟩} must be passed between braces.

wrapper = {⟨*code* {#1} *more code*⟩}   default: *empty*

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

6.6.3 The command `\printkeyans`

\printkeyans

`\printkeyans{`⟨*store name*⟩`}`
`\printkeyans[`⟨*keys*⟩`]{`⟨*store name*⟩`}`
`\printkeyans*[`⟨*keys*⟩`]{`⟨*store name*⟩`}`

The command `\printkeyans` prints *"all stored content"* in *sequence* {⟨*store name*⟩} defined by `save-ans` key placing this inside the `enumext` or `enumext*` environment if the *starred argument* '*' is used.

The *"stored content"* can only be accessed *after* it is stored in the *sequence*, if {⟨*store name*⟩} does not exist the command will return an error.

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

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

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

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

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

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

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

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

Example

```
\begin{enumext}[save-ans=sample,columns=1,show-pos=true,nosep,save-ref=true]
  \item Factor $3x+3y+3z$. \anskey{$3(x+y+z)$}
  \item True False

    \begin{enumext}[nosep]
      \item \LaTeX2e\ is cool? \anskey{Very True!}
    \end{enumext}

  \item Related to Linux

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

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

\printkeyans{sample}
```

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

[1] | $3(x + y + z)$

2. True False

    (a) LATEX2e is cool?

    [2] | Very True!

3. Related to Linux

    (a) You use Linux?

    [3] | Yes

    (b) Rate the following package and class

        i.   `xsim`
        [4] | very good

        ii. `exsheets`
        [5] | obsolete

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

1. $3(x + y + z)$ ※
2. (a) Very True! ※
3. (a) Yes ※
    (b) i.   very good ※
        ii. obsolete ※

7 Full examples

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

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

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

Example 1

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

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

    A 36 km/h.
    B 360 km/h.
    C 27,8 km/h.
    D $3,60 \times 10^8$ km/h.

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

    A $1 \text{ Å} = 1 \times 10^5$ fm.
    B $1 \text{ Å} = 1 \times 10^{-5}$ fm.
    C $1 \text{ Å} = 1 \times 10^{-15}$ fm.
    D $1 \text{ Å} = 1 \times 10^3$ fm.

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

    A 36 km/h.
    B 360 km/h.
    C 27,8 km/h.
    D $3,60 \times 10^8$ km/h.

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

    A $1 \text{ Å} = 1 \times 10^5$ fm.
    B $1 \text{ Å} = 1 \times 10^{-5}$ fm.
    C $1 \text{ Å} = 1 \times 10^{-15}$ fm.
    D $1 \text{ Å} = 1 \times 10^3$ fm.

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

Example 2

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

---

[1] The cool TEX automation tool: https://www.ctan.org/pkg/arara

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

   A 36 km/h.
   ✓ B 360 km/h.
   C 27,8 km/h.
   D $3{,}60 \times 10^8$ km/h.

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

   ✓ A $1\,\text{Å} = 1 \times 10^5$ fm.
   B $1\,\text{Å} = 1 \times 10^{-5}$ fm.
   C $1\,\text{Å} = 1 \times 10^{-15}$ fm.
   D $1\,\text{Å} = 1 \times 10^3$ fm.

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

   A 36 km/h.
   ✓ B 360 km/h.
   C 27,8 km/h.
   D $3{,}60 \times 10^8$ km/h.

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

   ✓ A $1\,\text{Å} = 1 \times 10^5$ fm.
   B $1\,\text{Å} = 1 \times 10^{-5}$ fm.
   C $1\,\text{Å} = 1 \times 10^{-15}$ fm.
   D $1\,\text{Å} = 1 \times 10^3$ fm.

1. B   ※ 2. A   ※
3. B   ※ 4. A   ※

Example 3

A *"simple multiple choice"* test 📄.

1. First type of questions
   Ⓐ value   Ⓑ correct
   Ⓒ value   Ⓓ value

2. Second type of questions
   I. $2\alpha + 2\delta = 90°$
   II. $\alpha = \delta$
   III. $\angle EDF = 45°$
   Ⓐ I only   Ⓓ I and III only
   Ⓑ II only   Ⓔ I, II, and III
   Ⓒ I and II only

3. Third type of questions
   (1) $2\alpha + 2\delta = 90°$
   (2) $\angle EDF = 45°$
   Ⓐ value   Ⓓ value
   Ⓑ value   Ⓔ value
   Ⓒ value

4. Question with image and label below:

   Ⓐ   Ⓑ   Ⓒ   Ⓓ   Ⓔ

5. Question with image on right side:
   Ⓐ value
   Ⓑ value
   Ⓒ value
   Ⓓ correct
   Ⓔ value

Test keys
1. B, $x = 5$ ※ 4. E, A duck   ※
2. D   ※ 5. D, other note   ※
3. C, some note   ※

Example 4

A *"simple worksheet"* using ducks :) 📄.

🦆1 Factor $x^2 - 2x + 1$

🦆2 Factor $3x + 3y + 3z$

   The following questions need to be cuaqtified :)

🦆3 True False
   (a) $\alpha > \delta$
   (b) LaTeX2e is cool?

🦆4 Related to Linux
   (a) You use Linux?

(b) Usually uses the package manager?
(c) Rate the following package and class
  i. `xsim-exam`
  ii. `xsim`
  iii. `exsheets`

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

1. $(x-1)^2$ ※
2. $3(x+y+z)$ ※
3. (a) False   ※
 (b) Very True!   ※
4. (a) Yes   ※

 (b) Yes, `dnf`   ※
 (c) i. doesn't exist for now :(※
  ii. very good   ※
  iii. obsolete   ※

Example 5

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

1

Which choice best describes what happens in the passage?

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

2

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.

4

Which choice best describes what happens in the passage?

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

1. A)   2. C)   3. B)   4. D)

Example 6

Adapted from the response to Environment for enumerate environment 📄.

8.5a, KSC   **10.** sample

 A sample
✓ **B** answer
 C sample
 D sample

9.5a, KSC   **11.** sample

 A sample
 B sample
 C sample
✓ **D** answer

12. sample

 A sample
 B answer
 C sample
 D sample

13. sample

 A sample
 B sample
 C sample
 D answer

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

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

8 Tagged PDF examples

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

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

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

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

- The file `enumext-01.tex` contains the basic tests for the `enumext` and `enumext*` environments and the nesting between them plus the use of the `label`, `labelwidth`, `labelsep`, `ref`, `align` and `wrap-label` keys. Source file 📄 and *tagged* PDF 📄.
- The file `enumext-02.tex` contains the tests for the `enumext` and `enumext*` environments and the support for `minipage` and `multicols` environments using the keys `columns`, `columns-sep`, `mini-env`, `mini-right` and `\miniright` command. Source file 📄 and *tagged* PDF 📄.
- The file `enumext-03.tex` contains the tests for the `enumext` and `keyanspic` environments activated by the `save-ans` key together with the `save-sep` and `save-ref` keys and the `\printkeyans` command. Source file 📄 and *tagged* PDF 📄.
- The file `enumext-04.tex` contains the tests for the `\anskey` command and the `anskey*` environment activated by the `save-ans` key along with the `\getkeyans` and `\printkeyans` commands. Source file 📄 and *tagged* PDF 📄.
- The file `enumext-05.tex` contains the tests for the environments `keyans`, `keyans*` and `keyanspic` activated by the key `save-ans` together with the keys `no-store` and `show-ans` and the commands `\setenumext`, `\setenumextmeta`, `\printkeyans` and `\foreachkeyans`. Source file 📄 and *tagged* PDF 📄.
- The file `enumext-06.tex` contains the tests for the environments `enumext` and `enumext*` for *fake* `itemize` and `description`. Source file 📄 and *tagged* PDF 📄.
- The file `enumext-07.tex` contains the tests for starting the environments with `\setenumext{resume}`, the `\resetenumext` command and the `series`, `resume`, `resume*` and `reset` keys. Source file 📄 and *tagged* PDF 📄.

9 The way of non-enumerated lists

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

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

Fake `itemize` environment

Here we set the `label` key using the default settings in LaTeX 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
  - Second level item
    * Third level item
      · Fourth level item
- First level item

* First level item
  ◇ Second level item
    ◦ Third level item
      ⋆ Fourth level item
* First level item

Fake `description` environment

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

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

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

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

- The small space at the beginning of the *"unlabeled entry"* corresponds to `\labelsep` and can be removed using `\hspace{-\labelsep}` at the beginning of the line.

- When *tagged* PDF is active the default `description` style is NOT available due to the redefinition of `\makelabel` for the `align` key which uses `\makebox` in this case, meaning that `\item[⟨content⟩]` will not extend beyond `\labelwidth` which causes overlaps,

Description indented by label

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

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

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

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

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

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

SoMeThInG LoNg A much longer description. Lorem ipsum dolor sit amet, consectetuer 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, 2025.

[4] González, Pablo. "scontents - Stores LaTeX contents in memory or files". Available from CTAN, https://www.ctan.org/pkg/scontents, 2025.

[5] The LaTeX Project. "enumerate – Enumerate with redefinable labels". Available from CTAN, https://www.ctan.org/pkg/enumerate, 2025.

[6] Bezos, Javier. "Customizing lists with the enumitem package". Available from CTAN, https://www.ctan.org/pkg/enumitem, 2025.

[7] Berry, Karl. "LaTeX 2ε: An Unofficial Reference Manual". Available from CTAN, https://ctan.org/pkg/latex2e-help-texinfo, 2025.

[8] The LaTeX Project. "Extensive support for hypertext in LaTeX". Available from CTAN, https://www.ctan.org/pkg/hyperref, 2025.

[9] Burnol, Jean-François. "The footnotehyper package". Available from CTAN, https://www.ctan.org/pkg/footnotehyper, 2021.

[10] The LaTeX Project. "The expl3 package". Available from CTAN, https://www.ctan.org/pkg/l3kernel, 2025.

[11] The LaTeX Project. "The LaTeX3 Interfaces". Available from CTAN, https://www.ctan.org/pkg/l3kernel, 2025.

[12] The LaTeX Project. "The LaTeX 2_ε sources". Available from CTAN, https://ctan.org/tex-archive/macros/latex/base, 2025.

[13] The LaTeX Project. "LaTeX News, Issue 41, June 2025". Available from CTAN, https://ctan.org/tex-archive/macros/latex/base, 2025.

[14] The LaTeX Project. "LaTeX for authors current version". Available from CTAN, https://ctan.org/pkg/latex-base, 2025.

[15] Gundlach, Patrick. "The lua-visual-debug package". Available from CTAN, https://www.ctan.org/pkg/lua-visual-debug, 2023.

[16] Lemvig, Mogens. "The shortlst package". Available from CTAN, https://www.ctan.org/pkg/shortlst, 1998.

[17] Niederberger, Clemens. "tasks – Horizontally columned lists". Available from CTAN, https://www.ctan.org/pkg/tasks, 2022.

[18] Fischer, Ulrike. "tagpdf – LaTeX kernel code for PDF tagging". Available from CTAN, https://www.ctan.org/pkg/tagpdf, 2025.

[19] The LaTeX Project. "latex-lab – LaTeX laboratory". Available from CTAN, https://www.ctan.org/pkg/latex-lab, 2025.

[20] Mittelbach, Frank. "LaTeX's socket management". Available from CTAN, https://ctan.org/tex-archive/macros/latex/base, 2025.

11 Change history

v1.9 (ctan), 2025-11-01
- Update requirement to LaTeX release 2025-11-01.
- Move `\setcounter` to `\begin{list}{⟨arg one⟩}{⟨arg two⟩\setcounter}`.

v1.8 (ctan), 2025-10-04
- Replacing `\scantokens` (primitive) with `\tl_retokenize:n`.
- Cleanup warnings and details returned by expltools.

v1.7 (ctan), 2025-07-10
- Fixed `\setenumext{enumext*}{resume}`.
- Fixed bad interaction between `\setenumext` and the `resume` key.
- The behavior of the key `resume*` has been updated and documented.

v1.6 (ctan), 2025-07-04
- Syntax simplification for `\setenumextmeta`.
- Environments can be started with the key `resume` *without value*.
- Add `\resetenumext`, `reset` and `reset*` keys.
- The `resume`, `resume*` and `series` keys can now be set per level.
- Fixed bad interaction between `\printkeyans` and the `resume`, `resume*` keys.

v1.5 (ctan), 2025-06-11
- Replacing `\regex_match:` (deprecated) with `\regex_if_match:`.
- Add keys `beginpenalty`, `midpenalty` and `endpenalty`.

v1.4 (ctan), 2025-06-09
- Improved implementation of the `start` key for *tagged* PDF.
- Improved implementation of the `ref` key.
- Fixed the behavior of the `save-sep` key.
- Fixed the behavior of the `resume*` key.

v1.3 (ctan), 2025-06-01
- Removed dependency on the scontents package.
- The `anskey*` environment has been rewritten using the new `c-type` argument.

v1.2 (ctan), 2025-03-28
- Replace signature (prevent expansion for optional argument).
- Solve Inconsistent local/global assignment.

v1.1 (ctan), 2024-11-14
- Fixed implementation for `font` and `base-fix` keys.
- Added new keys for symbol marks.
- Update and improvements in the internal code.
- Adjustments in the documentation.

v1.0 (ctan), 2024-11-01
- First public release.

12 Index of Documentation

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

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 (LATEX3 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 LATEX to work correctly.

```
3 \NeedsTeXFormat{LaTeX2e}[2025-11-01]
```

Now declare the enumext package.

```
4 \ProvidesExplPackage {enumext} {2025-11-01} {1.9} {Enumerate exercise sheets}
```

Finally check if the multicol package are loaded, if not we load it.

```
5 \hook_gput_code:nnn {begindocument} {enumext}
6   {
7     \IfPackageLoadedTF { multicol }
8       {
9         \msg_info:nnn { enumext } { package-load } { multicol }
10      }
11      {
12        \msg_info:nnn { enumext } { package-not-load } { multicol }
13        \RequirePackage{multicol}[2025-06-05]
14      }
15  }
```

13.4 Definition of variables

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

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

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

```
16 \int_new:N  \l__enumext_level_int
17 \int_new:N  \l__enumext_level_h_int
18 \int_new:N  \l__enumext_anskey_level_int
19 \int_new:N  \l__enumext_keyans_level_int
20 \int_new:N  \l__enumext_keyans_level_h_int
21 \int_new:N  \l__enumext_keyans_pic_level_int
```

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

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

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

```
22 \bool_new:N \l__enumext_starred_bool
23 \bool_new:N \g__enumext_starred_bool
24 \bool_new:N \l__enumext_starred_first_bool
25 \bool_new:N \l__enumext_standar_bool
26 \bool_new:N \g__enumext_standar_bool
27 \bool_new:N \l__enumext_standar_first_bool
28 \bool_new:N \l__enumext_keyans_env_bool
29 \tl_new:N   \g__enumext_start_line_tl
30 \tl_new:N   \g__enumext_envir_name_tl
31 \tl_new:N   \l__enumext_envir_name_tl
```

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

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

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

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

```
32 \cs_set_protected:Npn \__enumext_tmp:n #1
33   {
34     \tl_new:c { l__enumext_counter_#1_tl }
35   }
36 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }
```

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

`\l__enumext_ref_key_arg_tl`
`\l__enumext_ref_the_count_tl`
`\l__enumext_renew_counter_X_tl`
`\l__enumext_the_counter_X_tl`

Internal variables used by ref key (§13.14).

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

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

`\l__enumext_series_name_tl`
`\l__enumext_resume_count_bool`
`\l__enumext_resume_count_X_bool`
`\l__enumext_resume_series_X_bool`
`\l__enumext_resume_star_key_X_bool`
`\g__enumext_resume_last_keys_X_tl`

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

```
46 \tl_new:N \l__enumext_series_name_tl
47 \bool_new:N \l__enumext_resume_count_bool
48 \cs_set_protected:Npn \__enumext_tmp:n #1
49   {
50     \bool_new:c { l__enumext_resume_count_#1_bool    }
51     \bool_new:c { l__enumext_resume_series_#1_bool   }
52     \bool_new:c { l__enumext_resume_star_key_#1_bool }
53     \tl_new:c   { g__enumext_resume_last_keys_#1_tl  }
54   }
55 \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {#1} }
```

(*End of definition for* `\l__enumext_series_name_tl` *and others.*)

`\l__enumext_current_widest_dim`
`\g__enumext_counter_styles_tl`
`\g__enumext_widest_label_tl`
`\l__enumext_label_width_by_box`

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

```
56 \dim_new:N \l__enumext_current_widest_dim
57 \tl_new:N  \g__enumext_counter_styles_tl
58 \tl_new:N  \g__enumext_widest_label_tl
59 \box_new:N \l__enumext_label_width_by_box
```

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

The boolean variable `\l__enumext_leftmargin_tmp_X_bool` and the dimensional variable `\l__enumext_-leftmargin_tmp_X_dim` are used by the `list-indent` key (§13.19). The variables `\l__enumext_-leftmargin_X_dim` and `\l__enumext_itemindent_X_dim` are used and set by the function `\__enumext_-calc_hspace:NNNNNNNNNNNN` (§13.41.1).

```
60  \cs_set_protected:Npn \__enumext_tmp:n #1
61    {
62      \bool_new:c { l__enumext_leftmargin_tmp_#1_bool }
63      \dim_new:c  { l__enumext_leftmargin_tmp_#1_dim  }
64      \dim_new:c  { l__enumext_leftmargin_#1_dim      }
65      \dim_new:c  { l__enumext_itemindent_#1_dim      }
66    }
67  \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_leftmargin_tmp_X_bool`
`\l__enumext_leftmargin_tmp_X_dim`
`\l__enumext_leftmargin_X_dim`
`\l__enumext_itemindent_X_dim`

Internal variables used by `columns` key (§13.23) and `align` key (§13.13).

```
68  \cs_set_protected:Npn \__enumext_tmp:n #1
69    {
70      \skip_new:c  { l__enumext_multicols_above_#1_skip }
71      \skip_new:c  { l__enumext_multicols_below_#1_skip }
72      \skip_new:c  { g__enumext_multicols_right_#1_skip }
73      \str_new:c   { l__enumext_align_label_pos_#1_str  }
74    }
75  \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.*)

`\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 `\miniright` command (§13.24.4) and the keys `mini-right`, `mini-right*`, `mini-env` and `mini-sep` (§13.22, §13.24).

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

`\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`

The bool vars `\l__enumext_wrap_label_X_bool` and `\l__enumext_wrap_label_opt_X_bool` are used by `wrap-label` and `wrap-label*` keys (§13.13), the integer `\l__enumext_start_X_int` are used by the `start` and `start*` keys (§13.15), the token list `\l__enumext_fake_item_indent_X_tl` is used by `itemindent` key (§13.19.1), the variables `\l__enumext_label_fill_left_X_tl` and `\l__enumext_-label_fill_left_X_tl` are used by the `align` key (§13.13). The boolean vars `\l__enumext_vspace_-a_star_X_bool`, `\l__enumext_vspace_b_star_X_bool` are used by `above`, `above*`, `below` and `below*` keys (§13.21).

```
89   \cs_set_protected:Npn \__enumext_tmp:n #1
90     {
91       \bool_new:c { l__enumext_wrap_label_#1_bool      }
92       \bool_new:c { l__enumext_wrap_label_opt_#1_bool  }
93       \int_new:c  { l__enumext_start_#1_int            }
94       \tl_new:c   { l__enumext_fake_item_indent_#1_tl  }
95       \tl_new:c   { l__enumext_label_fill_left_#1_tl   }
96       \tl_new:c   { l__enumext_label_fill_right_#1_tl  }
97       \bool_new:c { l__enumext_vspace_a_star_#1_bool   }
98       \bool_new:c { l__enumext_vspace_b_star_#1_bool   }
99     }
100  \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_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 variable `\l__enumext_store_active_bool` setting by `save-ans` key (§13.29.1) activates all the mechanism related to `\anskey`, `anskey*`, `keyans`, `keyans*` and `keyanspic` environments.

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

The variables `\l__enumext_store_current_label_tl` and `\l__enumext_store_current_opt_arg_tl` save the *current label* and *optional argument* of `\item*` (§13.40) and `\anspic*` (§13.45.2) for the `keyans`, `keyans*` and `keyanspic` environments.

```
101 \bool_new:N \l__enumext_store_active_bool
102 \tl_new:N   \l__enumext_store_name_tl
103 \tl_new:N   \g__enumext_store_name_tl
104 \tl_new:N   \l__enumext_store_current_label_tl
105 \tl_new:N   \l__enumext_store_current_opt_arg_tl
```

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

The variable `\l__enumext_store_anskey_arg_tl` save the *argument* of `\anskey` (§13.33) and the variables `\l__enumext_store_anskey_env_tl` save the ⟨*body*⟩ of the environment `anskey*` (§13.34).

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

```
106 \tl_new:N   \l__enumext_store_anskey_arg_tl
107 \tl_new:N   \l__enumext_store_anskey_env_tl
108 \bool_new:N \l__enumext_write_anskey_env_bool
109 \tl_new:N   \l__enumext_write_anskey_env_file_name_tl
110 \iow_new:N  \l__enumext_write_anskey_env_file_iow
```

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

The `\c__enumext_anskey_env_hidden_space_str` is a constant *string* to used to hide the ⟨*forced space*⟩ added by TeX when recording content in a macro. This *string* contains the *reserved phrase* '`%^^Aenumextheol%`' which is added to the end of the argument stored in *sequence* and *prop list* when the key `force-eol` is false.

```
111 \str_const:Ne \c__enumext_anskey_env_hidden_space_str
112   { \c_percent_str \c_circumflex_str \c_circumflex_str A enumextheol \c_percent_str }
```

(*End of definition for* `\c__enumext_anskey_env_hidden_space_str`.)

Internal variables used by the command `\setenumext` (§13.51).

```
113 \tl_new:N  \l__enumext_setkey_tmpa_tl
114 \tl_new:N  \l__enumext_setkey_tmpb_tl
115 \int_new:N \l__enumext_setkey_tmpa_int
116 \seq_new:N \l__enumext_setkey_tmpa_seq
117 \seq_new:N \l__enumext_setkey_tmpb_seq
```

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

Internal variables used by the `\setenumextmeta` command (§13.52) and `\foreachkeyans` command (§13.53).

```
118 \str_new:N \l__enumext_meta_path_str
119 \seq_new:N \l__enumext_foreach_print_seq
120 \tl_new:N  \l__enumext_foreach_name_prop_tl
121 \tl_new:N  \l__enumext_foreach_default_keys_tl
```

(*End of definition for* `\l__enumext_meta_path_str` *and others.*)

Internal variables used by command `\printkeyans` (§13.50), `show-pos`, `show-ans`, `mark-pos`, `mark-sep` keys (§13.30), `item-sym*` key (§13.38), `save-key` key (§13.30.3) and *"storing structure"*.

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

```
135        \tl_new:c   { l__enumext_store_save_key_#1_tl       }
136        \bool_new:c { l__enumext_store_save_key_#1_bool     }
137        \bool_new:c { l__enumext_store_upper_level_#1_bool }
138      }
139 \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {#1} }
```

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

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

Internal variables used by keyanspic environment and \anspic command (§13.45.1).

```
140 \seq_new:N  \l__enumext_anspic_args_seq
141 \dim_new:N  \l__enumext_anspic_mini_width_dim
142 \int_new:N  \l__enumext_anspic_above_int
143 \int_new:N  \l__enumext_anspic_below_int
144 \bool_new:N \l__enumext_anspic_label_above_bool
145 \str_new:N  \l__enumext_anspic_mini_pos_str
146 \box_new:N  \l__enumext_anspic_label_box
147 \box_new:N  \l__enumext_anspic_body_box
148 \dim_new:N  \l__enumext_anspic_label_htdp_dim
149 \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
\l__enumext_item_wrap_key_bool
\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 (§13.29.3) used by the check-ans, no-store, wrap-ans* keys and check for starred commands \item* in keyans and keyans* environments and \anspic* in keyanspic environment.

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

```
159 \bool_new:N \l__enumext_hyperref_bool
160 \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 (§13.30). 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.

```
161 \tl_new:N \l__enumext_newlabel_arg_one_tl
162 \tl_new:N \l__enumext_newlabel_arg_two_tl
163 \tl_new:N \l__enumext_write_aux_file_tl
164 \cs_set_protected:Npn \__enumext_tmp:n #1
165   {
166     \tl_new:c { l__enumext_label_copy_#1_tl }
167   }
168 \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).

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

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

Internal variables used by enumext* and keyans* environments.

```
175 \cs_set_protected:Npn \__enumext_tmp:n #1
176   {
177     \bool_new:c { l__enumext_item_starred_#1_bool    }
178     \int_new:c  { l__enumext_item_column_pos_#1_int  }
179     \int_new:c  { g__enumext_item_count_all_#1_int   }
180     \int_new:c  { l__enumext_joined_item_#1_int      }
181     \int_new:c  { l__enumext_joined_item_aux_#1_int  }
182     \int_new:c  { l__enumext_tmpa_#1_int             }
183     \dim_new:c  { l__enumext_tmpa_#1_dim             }
184     \box_new:c  { l__enumext_item_text_#1_box        }
185     \dim_new:c  { l__enumext_joined_width_#1_dim     }
186     \dim_new:c  { l__enumext_item_width_#1_dim       }
187     \tl_new:c   { g__enumext_item_symbol_aux_#1_tl   }
188     \str_new:c  { l__enumext_align_label_#1_str      }
189     \bool_new:c { g__enumext_minipage_active_#1_bool }
190     \box_new:c  { l__enumext_miniright_code_#1_box   }
191     \bool_new:c { g__enumext_minipage_center_#1_bool }
192     \dim_new:c  { g__enumext_minipage_right_#1_dim   }
193     \skip_new:c { g__enumext_minipage_right_#1_skip  }
194   }
195 \clist_map_inline:nn { vii, viii } { \__enumext_tmp:n {#1} }
```

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

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

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

(*End of definition for* \c__enumext_all_envs_clist.)

13.5 Some utility functions

Non-standard kernel variants used by the \printkeyans command (§13.50) and \foreachkeyans command (§13.53).

```
201 \cs_generate_variant:Nn \keys_precompile:nnN { neN }
202 \cs_generate_variant:Nn \seq_use:Nn { NV }
```

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

A internal *"hook"* function used for copying plain list and minipage environments definition and hyperref detection.

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

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

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.

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

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

Function for check current level in enumext.

```
215 \cs_new:Nn \__enumext_level:
216   {
217     \int_to_roman:n { \l__enumext_level_int }
218   }
```

(*End of definition for* `\__enumext_level:`.)

`\__enumext_if_is_int:nT`
`\__enumext_if_is_int:nF`
`\__enumext_if_is_int:nTF`

A conditional function to determine whether the variable we are passing is an integer used by `start` and `widest` keys. This function is taken directly from the answer given by Henri Menke in How to test if an expl3 function argument is an integer expression?.

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

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

`\__enumext_show_length:nnn`

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

```
225  \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
226    {
227      *~#2
228      \prg_replicate:nn { 14 - \str_count:n {#2} } {~}
229        =~\use:c { #1_use:c } { l__enumext_#2_#3_#1 } \\
230    }
```

(*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` depending on the case.

```
231  \cs_new_protected:Nn \__enumext_unskip_unkern:
232    {
233      \int_case:nnT { \lastnodetype }
234        {
235          { 11 }{ \unskip }
236          { 12 }{ \unkern }
237        }
238    }
```

(*End of definition for* `\__enumext_unskip_unkern:`.)

13.5.1 Utilities for environments and levels

`\__enumext_is_not_nested:`
`\__enumext_is_on_first_level:`

The function `\__enumext_is_not_nested:` sets 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 saves the environment name in `\l__enumext_envir_name_tl`.

```
239  \cs_new_protected:Nn \__enumext_is_not_nested:
240    {
241      \str_case:en { \@currenvir }
242        {
243          {enumext}
244            {
245              \tl_set:Nn \l__enumext_envir_name_tl { enumext }
246              \bool_lazy_and:nnT
247                { \bool_not_p:n { \g__enumext_standar_bool } }
248                { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
249                {
250                  \bool_gset_true:N \g__enumext_standar_bool
251                }
252            }
253          {enumext*}
254            {
255              \tl_set:Nn \l__enumext_envir_name_tl { enumext* }
256              \bool_lazy_and:nnT
257                { \bool_not_p:n { \g__enumext_starred_bool } }
258                { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
259                {
260                  \bool_gset_true:N \g__enumext_starred_bool
261                }
262            }
263        }
264    }
```

The function `\__enumext_is_on_first_level:` will sets the variables `\l__enumext_standar_first_-bool` (§13.29.1), `\l__enumext_starred_first_bool` (§13.29.1) to *"true"* only if the environment is not nested and we are in the *"first level"* of it . We will also save the *start line number* of each environment in the variable `\g__enumext_start_line_tl` and the *name* of each environment in the variable `\g__enumext_-envir_name_tl` to use in messages related to the check-ans key and .log file.

```
265  \cs_new_protected:Nn \__enumext_is_on_first_level:
266    {
267      \bool_lazy_all:nT
268        {
269          { \bool_if_p:N \g__enumext_standar_bool }
270          { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
271          { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
272        }
273        {
274          \bool_set_true:N \l__enumext_standar_first_bool
275          \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
276          \tl_gset:Ne \g__enumext_start_line_tl
277            {
278              on~line~\exp_not:V \inputlineno
279            }
280        }
281      \bool_lazy_all:nT
282        {
283          { \bool_if_p:N \g__enumext_starred_bool }
284          { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
285          { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
286        }
287        {
288          \bool_set_true:N \l__enumext_starred_first_bool
289          \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
290          \tl_gset:Ne \g__enumext_start_line_tl
291            {
292              on~line~\exp_not:V \inputlineno
293            }
294        }
295    }
```

(*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.

```
296  \cs_new_protected:Nn \__enumext_keyans_name_and_start:
297    {
298      \str_case:en { \@currenvir }
299        {
300          {keyans}
301            {
302              \tl_set:Nn \l__enumext_envir_name_tl { keyans }
303              \tl_set:Ne \l__enumext_check_start_line_env_tl
304                {
305                  in~'keyans'~start~on~line~\exp_not:V \inputlineno
306                }
307            }
308          {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          {keyanspic}
317            {
318              \tl_set:Nn \l__enumext_envir_name_tl { keyanspic }
319              \tl_set:Ne \l__enumext_check_start_line_env_tl
320                {
321                  in~'keyanspic'~start~on~line~\exp_not:V \inputlineno
322                }
323            }
```

```
324        }
325      }
```

(*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.

```
326 \cs_new_protected:Nn \__enumext_reset_global_vars:
327   {
328     \__enumext_reset_global_int:
329     \__enumext_reset_global_bool:
330     \__enumext_reset_global_tl:
331   }
332 \cs_new_protected:Nn \__enumext_reset_global_int:
333   {
334     \int_gzero:N \g__enumext_item_number_int
335     \int_gzero:N \g__enumext_item_anskey_int
336     \int_gzero:N \g__enumext_item_answer_diff_int
337   }
338 \cs_new_protected:Nn \__enumext_reset_global_bool:
339   {
340     \bool_gset_false:N \g__enumext_check_ans_key_bool
341     \bool_gset_false:N \g__enumext_standar_bool
342     \bool_gset_false:N \g__enumext_starred_bool
343   }
344 \cs_new_protected:Nn \__enumext_reset_global_tl:
345   {
346     \tl_gclear:N \g__enumext_store_name_tl
347     \tl_gclear:N \g__enumext_start_line_tl
348     \tl_gclear:N \g__enumext_envir_name_tl
349   }
```

(*End of definition for* __enumext_reset_global_vars: *and others.*)

__enumext_log_global_vars:
__enumext_log_answer_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.

```
350 \cs_new_protected:Nn \__enumext_log_global_vars:
351   {
352     \msg_log:nneeee { enumext } { prop-seq-int-hook }
353       { \g__enumext_store_name_tl }
354       { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
355       { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
356       { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
357   }
```

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.

```
358 \cs_new_protected:Nn \__enumext_log_answer_vars:
359   {
360     \msg_log:nneee { enumext } { item-answer-hook }
361       { \int_use:N \g__enumext_item_number_int }
362       { \int_use:N \g__enumext_item_anskey_int }
363       { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int} }
364   }
```

(*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 ltcmd (see latex-lab-block[19]).

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

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

```
365 \__enumext_at_begin_document:n
366   {
367     \cs_new_eq:NN   \__enumext_start_list:nn \list
368     \cs_new_eq:NN   \__enumext_stop_list: \endlist
369     \NewCommandCopy \__enumext_item_std:w \item
370   }
```

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

```
371 \__enumext_at_begin_document:n
372   {
373     \cs_new_eq:NN \__enumext_minipage:w \minipage
374     \cs_new_eq:NN \__enumext_endminipage: \endminipage
375   }
```

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

13.7   Compatibility with hyperref and footnotehyper

__enumext_after_hyperref:
__enumext_hypertarget:nn
__enumext_phantomsection:

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

```
376 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
377 \hook_gset_rule:nnnn { begindocument } { enumext } { after } { hyperref }
```

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

```
378 \cs_new_protected:Nn \__enumext_after_hyperref:
379   {
380     \IfPackageLoadedT { hyperref }
381       {
382         \msg_info:nnn { enumext } { package-load } { hyperref }
383         \bool_set_true:N \l__enumext_hyperref_bool
384         \IfHyperBoolean{hyperfootnotes}
385           {
386             \bool_set_true:N \l__enumext_footnotes_key_bool
387           }
388           {  }
389       }
```

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.

```
390     \bool_if:NT \l__enumext_footnotes_key_bool
391       {
392         \IfPackageLoadedTF { footnotehyper }
393           {
394             \msg_info:nnn { enumext } { package-load } { footnotehyper }
395           }
396           {
397             \bool_set_false:N \l__enumext_footnotes_key_bool
398           }
399       }
```

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

```
400     \bool_if:NTF \l__enumext_hyperref_bool
401       {
402         \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
403         \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
404       }
405       {
406         \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
```

```
407        \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
408      }
409    }
```

(*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.

```
410  \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
411    {
412      \protected@write \@auxout { }
413        {
414          \token_to_str:N \newlabel {#1}
415            {
416              {#2}
417              \bool_if:NT \l__enumext_hyperref_bool
418                { { \thepage } {#2} {#1} }
419              { }
420            }
421        }
422      \__enumext_hypertarget:nn {#1} { }
423      \__enumext_phantomsection:
424    }
```

(*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.

```
425  \cs_new_protected:Nn \__enumext_footnotetext:nn
426    {
427      \footnotetext[#1]{#2}
428    }
429  \cs_new_protected:Nn \__enumext_renew_footnote:
430    {
431      \RenewDocumentCommand \footnote { o +m }
432        {
433          \tl_if_novalue:nTF {##1}
434            {
435              \stepcounter{footnote}
436              \int_gset_eq:Nc \g__enumext_footnote_standar_int { c@footnote }
437            }
438            {
439              \int_gset:Nn \g__enumext_footnote_standar_int { ##1 }
440            }
441          \footnotemark [ \g__enumext_footnote_standar_int ]
442          \seq_gput_right:Nn \g__enumext_footnote_standar_arg_seq { ##2 }
443          \seq_gput_right:NV
444            \g__enumext_footnote_standar_int_seq \g__enumext_footnote_standar_int
445        }
446    }
447  \cs_new_protected:Nn \__enumext_print_footnote:
448    {
449      \seq_if_empty:NF \g__enumext_footnote_standar_int_seq
450        {
451          \seq_map_pairwise_function:NNN
452            \g__enumext_footnote_standar_int_seq
453            \g__enumext_footnote_standar_arg_seq
454            \__enumext_footnotetext:nn
455        }
456      \seq_gclear:N \g__enumext_footnote_standar_arg_seq
457      \seq_gclear:N \g__enumext_footnote_standar_int_seq
458    }
```

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.

```
459  \cs_new_protected:Nn \__enumext_renew_footnote_mini:
460    {
461      \RenewDocumentCommand \footnote { o +m }
462        {
463          \tl_if_novalue:nTF {##1}
464            {
465              \stepcounter{footnote}
466              \int_gset_eq:Nc \g__enumext_footnote_starred_int { c@footnote }
467            }
468            {
469              \int_gset:Nn \g__enumext_footnote_starred_int { ##1 }
470            }
471          \footnotemark [ \g__enumext_footnote_starred_int ]
472          \seq_gput_right:Nn \g__enumext_footnote_starred_arg_seq { ##2 }
473          \seq_gput_right:NV
474            \g__enumext_footnote_starred_int_seq \g__enumext_footnote_starred_int
475        }
476    }
477  \cs_new_protected:Nn \__enumext_print_footnote_mini:
478    {
479      \seq_if_empty:NF \g__enumext_footnote_starred_int_seq
480        {
481          \seq_map_pairwise_function:NNN
482            \g__enumext_footnote_starred_int_seq
483            \g__enumext_footnote_starred_arg_seq
484            \__enumext_footnotetext:nn
485        }
486      \seq_gclear:N \g__enumext_footnote_starred_arg_seq
487      \seq_gclear:N \g__enumext_footnote_starred_int_seq
488    }
```

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

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

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

```
489  \cs_new_protected:Nn \__enumext_renew_footnote_standar:
490    {
491      \bool_if:NT \g__enumext_standar_bool
492        {
493          \IfDocumentMetadataTF
494            {
495              \__enumext_renew_footnote:
496            }
497            {
498              \bool_if:NF \l__enumext_footnotes_key_bool
499                {
500                  \__enumext_renew_footnote:
501                }
502            }
503        }
504    }
505  \cs_new_protected:Nn \__enumext_print_footnote_standar:
506    {
507      \bool_if:NT \g__enumext_standar_bool
508        {
509          \IfDocumentMetadataTF
510            {
511              \__enumext_print_footnote:
512            }
513            {
514              \bool_if:NF \l__enumext_footnotes_key_bool
515                {
516                  \__enumext_print_footnote:
517                }
518            }
519        }
520    }
```

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.

```
521  \cs_new_protected:Nn \__enumext_renew_footnote_starred:
522    {
523      \IfDocumentMetadataTF
524        {
525          \__enumext_renew_footnote_mini:
526        }
527        {
528          \bool_if:NF \l__enumext_footnotes_key_bool
529            {
530              \__enumext_renew_footnote_mini:
531            }
532        }
533    }
534  \cs_new_protected:Nn \__enumext_print_footnote_starred:
535    {
536      \IfDocumentMetadataTF
537        {
538          \__enumext_print_footnote_mini:
539        }
540        {
541          \bool_if:NF \l__enumext_footnotes_key_bool
542            {
543              \__enumext_print_footnote_mini:
544            }
545        }
546    }
```

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

```
547  \__enumext_after_env:nn { enumext* }
548    {
549      \__enumext_print_footnote_starred:
550    }
551  \__enumext_after_env:nn { keyans* }
552    {
553      \__enumext_print_footnote_starred:
554    }
```

(*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 implementation is adapted from the answer given by Max Chernoff (@MaxChernoff) in Customize minipage to support spaces below it.

💣 This function is passed to the function __enumext_safe_exec: in the enumext environment definition (§13.42) and __enumext_safe_exec_vii: in the enumext* environment definition (§13.47).

```
555  \cs_new_protected:Nn \__enumext_internal_mini_page:
556    {
557      \int_compare:nNnT { \l__enumext_level_int } = { 0 }
558        {
559          \DeclareDocumentEnvironment{__enumext_mini_page}{ m }
560            {
561              \__enumext_renew_footnote_standar:
562              \__enumext_minipage:w [ t ] { ##1 }
563                \legacy_if_gset_false:n { @minipage }
564                \skip_vertical:N \c_zero_skip
565            }
566            {
567                \skip_vertical:N \c_zero_skip
568              \__enumext_endminipage:
569              \__enumext_print_footnote_standar:
570            }
571        }
```

```
572    }
```

(*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.

```
573 \dim_zero_new:N \itemwidth
```

13.11   Definition of counters

`\__enumext_define_counter: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.

```
574 \cs_new_protected:Npn \__enumext_define_counter:Nn #1 #2
575   {
576     \cs_if_exist:cTF { c@ #2 }
577       { \msg_fatal:nnn { enumext } { counters }{ #2 } }
578       {
579         \tl_set:Nn #1 { #2 }
580         \newcounter { #2 }
581       }
582   }
```

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.

```
583 \__enumext_define_counter:Nn \l__enumext_counter_i_tl     { enumXi     }
584 \__enumext_define_counter:Nn \l__enumext_counter_ii_tl    { enumXii    }
585 \__enumext_define_counter:Nn \l__enumext_counter_iii_tl   { enumXiii   }
586 \__enumext_define_counter:Nn \l__enumext_counter_iv_tl    { enumXiv    }
587 \__enumext_define_counter:Nn \l__enumext_counter_v_tl     { enumXv     }
588 \__enumext_define_counter:Nn \l__enumext_counter_vi_tl    { enumXvi    }
589 \__enumext_define_counter:Nn \l__enumext_counter_vii_tl   { enumXvii   }
590 \__enumext_define_counter:Nn \l__enumext_counter_viii_tl  { enumXviii  }
```

(*End of definition for* `\__enumext_define_counter:Nn` *and others.*)

`\c@__enumext_resume_i_int`
`\c@__enumext_resume_ii_int`
`\c@__enumext_resume_iii_int`
`\c@__enumext_resume_iv_int`
`\c@__enumext_resume_vii_int`

In version 1.6 the command `\resetenumext` (§13.27) was added which internally uses `\counterwithin*` so for its correct operation, we will create *"real counters"* instead of the *"integer variables"* for the keys resume and resume*.

```
591 \cs_set_protected:Npn \__enumext_tmp:n #1
592   {
593     \cs_if_exist:cTF { c@ __enumext_resume_#1_int }
594       { \msg_fatal:nne { enumext } { counters }{ __enumext_resume_#1_int } }
595       {
596         \newcounter { __enumext_resume_#1_int }
597       }
598   }
599 \clist_map_inline:nn {i,ii,iii,iv,vii} { \__enumext_tmp:n {#1} }
```

(*End of definition for* `\c@__enumext_resume_i_int` *and others.*)

13.12   Definition of labels

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

⚓ Direct support for this is provided since LaTeX release 2025-06-01[13], but we will keep the original implementation so as not to hinder the internal *"label and ref"* system.

`\__enumext_register_default_label_wd: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_default_label_wd: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_default_label_wd:Nn \arabic { 0 }
606  \__enumext_register_default_label_wd:Nn \Alph   { M }
607  \__enumext_register_default_label_wd:Nn \alph   { m }
608  \__enumext_register_default_label_wd:Nn \Roman  { VIII }
609  \__enumext_register_default_label_wd:Nn \roman  { viii }
```

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

__enumext_label_width_by_box:Nn
__enumext_label_width_by_box:cv

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

```
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 \l__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 (§13.37). 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 = \l__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
labelsep
labelwidth
wrap-label
wrap-label*

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   = { l__enumext_label_font_style_#2_tl },
```

```
647          font        .value_required:n = true,
648          labelsep    .dim_set:c  = { l__enumext_labelsep_#2_dim },
649          labelsep    .initial:n  = {0.3333em},
650          labelsep    .value_required:n = true,
651          labelwidth  .dim_set:c  = { l__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 { l__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 `\l__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:nneee { 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:nneee { 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 ⟨*label*⟩, the value of the variables \l__enumext_label_X_tl, the default values for \labelwidth and the *"label and ref"* system.

13.14.1 Define and set label and ref keys for enumext environment

label
ref
\l__enumext_label_i_tl
\l__enumext_label_ii_tl
\l__enumext_label_iii_tl
\l__enumext_label_iv_tl

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

```
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:cvn { 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
__enumext_standar_ref:

The __enumext_standard_ref:n function will first pass the key *argument* ref to the variable \l__enumext_ref_key_arg_tl and analyze its state, if it is not *empty* it will set a copy of of the *current counter style* save in \l__enumext_the_counter_X_tl to \l__enumext_ref_the_count_tl and then set the variable \l__enumext_renew_counter_X_tl which will modify \theenumX.

```
735  \cs_new_protected:Npn \__enumext_standar_ref:n #1
736    {
737      \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
738      \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
739        {
740          \msg_error:nnn { enumext } { key-ref-empty } { enumext }
741        }
742        {
743          \tl_set_eq:Nc \l__enumext_ref_the_count_tl
744            {
745              l__enumext_the_counter_ \__enumext_level: _tl
746            }
747          \tl_set:ce { l__enumext_renew_counter_ \__enumext_level: _tl }
748            {
749              \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
750                { \exp_not:V \l__enumext_ref_key_arg_tl }
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 { l__enumext_renew_counter_ \__enumext_level: _tl }
757        {
758          \tl_use:c { l__enumext_renew_counter_ \__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

ref

\l__enumext_label_vii_tl

\l__enumext_label_viii_tl

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

```
761 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
762   {
763     \keys_define:nn { enumext / #1 }
764       {
765         label .code:n    = {
766                             \__enumext_label_style:cvn { l__enumext_label_#2_tl }
767                               { l__enumext_counter_#2_tl } {##1}
768                             \dim_set_eq:cN  { l__enumext_labelwidth_#2_dim }
769                               \l__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

__enumext_starred_ref:

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_the_counter_vii_tl
790             \tl_set:Ne \l__enumext_renew_counter_vii_tl
791               {
792                 \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V
793               }
794           }
795       }
796     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
797       {
798         \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
799           {
800             \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
801           }
802           {
803             \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
804             \tl_set:Ne \l__enumext_renew_counter_viii_tl
805               {
806                 \exp_not:N \renewcommand  { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:\
807               }
808           }
809       }
810   }
```

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.

```
811 \cs_new_protected:Nn \__enumext_starred_ref:
812   {
813     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
814       {
815         \tl_if_empty:NF \l__enumext_renew_counter_vii_tl
816           {
817             \tl_use:N \l__enumext_renew_counter_vii_tl
818           }
819       }
820     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
821       {
```

```
822                \tl_if_empty:NF \l__enumext_renew_counter_viii_tl
823                  {
824                    \tl_use:N \l__enumext_renew_counter_viii_tl
825                  }
826              }
827          }
```

(*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

label
ref
\l__enumext_label_v_tl
\l__enumext_label_vi_tl

Here we set the default ⟨*label*⟩ for `keyans` and `keyanspic` environment, along with the default value for `labelwidth` if it has not been established and `ref` key. The `keyanspic` environment use the same ⟨*label*⟩ as the `keyans` environment.

```
828 \keys_define:nn { enumext / keyans }
829  {
830    label .code:n    = {
831                        \__enumext_label_style:cvn { l__enumext_label_v_tl }
832                          { l__enumext_counter_v_tl } {#1}
833                        \__enumext_label_style:cvn { l__enumext_label_vi_tl }
834                          { l__enumext_counter_vi_tl } {#1}
835                        \dim_set_eq:NN
836                          \l__enumext_labelwidth_v_dim \l__enumext_current_widest_dim
837                      },
838    label .initial:n = \Alph*),
839    label .value_required:n = true,
840    ref    .code:n    = \__enumext_keyans_ref:n {#1},
841    ref    .value_required:n = true,
842  }
```

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

__enumext_keyans_ref:n
__enumext_keyans_ref:

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

```
843 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
844  {
845    \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
846    \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
847      {
848        \msg_error:nnn { enumext } { key-ref-empty } { keyans }
849      }
850      {
851        \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
852        \tl_put_right:Ne \l__enumext_renew_counter_v_tl
853          {
854            \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V \l__
855          }
856      }
857  }
```

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

```
858 \cs_new_protected:Nn \__enumext_keyans_ref:
859  {
860    \tl_if_empty:NF \l__enumext_renew_counter_v_tl
861      {
862        \tl_use:N \l__enumext_renew_counter_v_tl
863      }
864  }
```

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

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

__enumext_start_from:NNn
__enumext_start_from:ccn
__enumext_start_from:cce

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

#1: \l__enumext_label_X_tl
#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.

🌶️ In version 1.6 it is allowed to pass the `resume` key *without value* by means of the command `\setenumext`, for the correct operation of this we must set the boolean variable `\l__enumext_resume_count_bool` set by the `resume` key *without value* to *"false"* (§13.26). This is necessary to be able to "reset" the *start value* by means of the `start` or `start*` keys.

```
865  \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
866    {
867      \bool_set_false:N \l__enumext_resume_count_bool
868      \__enumext_if_is_int:nTF { #3 }
869        {
870          \int_set:Nn #2 {#3}
871        }
872        {
873          \regex_if_match:nVT { \c{Alph} | \c{alph} } #1
874            { \int_set:Nn #2 { \int_from_alph:n {#3} } }
875          \regex_if_match:nVT { \c{Roman} | \c{roman} } #1
876            { \int_set:Nn #2  { \int_from_roman:n {#3} } }
877        }
878    }
879  \cs_generate_variant:Nn \__enumext_start_from:NNn { ccn, cce }
```

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

`\__enumext_widest_from:nNNn`
`\__enumext_widest_from:nccn`

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

#1 : The counter associated with the environment level
#2 : `\l__enumext_label_X_tl`
#3 : `\l__enumext_labelwidth_X_dim`
#4 : ⟨*integer or string*⟩

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

```
880  \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
881    {
882      \__enumext_if_is_int:nTF {#4}
883        {
884          \setcounter{enumX#1} { #4 }
885        }
886        {
887          \regex_if_match:nVT { \c{Alph} | \c{alph} } #2
888            { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
889          \regex_if_match:nVT { \c{Alph} | \c{alph} } #2
890            { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
891        }
892      \__enumext_label_width_by_box:cv
893        { l__enumext_labelwidth_#1_dim } { l__enumext_label_#1_tl }
894    }
895  \cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }
```

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

`start`
`start*`
`widest`

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

```
896  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
897    {
898      \keys_define:nn { enumext / #1 }
899        {
900          start* .code:n     = {
901                                \__enumext_start_from:ccn
902                                  { l__enumext_label_#2_tl }
903                                  { l__enumext_start_#2_int } {##1}
904                               },
905          start* .value_required:n = true,
906          start  .code:n     = {
907                                \__enumext_start_from:cce
908                                  { l__enumext_label_#2_tl }
909                                  { l__enumext_start_#2_int } { \int_eval:n {##1} }
910                               },
911          start  .initial:n = 1,
912          start  .value_required:n = true,
913          widest .code:n     = {
```

```
914                              \__enumext_widest_from:nccn {#2}
915                                { l__enumext_label_#2_tl }
916                                { l__enumext_labelwidth_#2_dim } {##1}
917                              },
918              widest .value_required:n = true,
919          }
920      }
921  \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

13.16 Setting keys for penaltys

beginpenalty The three parameters \@beginparpenalty, \@itempenalty and \@endparpenalty work together to en-
midpenalty sure that list environments look good, avoiding unsightly page breaks that can break the flow of the list, so
endpenalty it's a good idea to have a ⟨keys⟩ to access these.

```
922  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
923    {
924      \keys_define:nn { enumext / #1 }
925        {
926          beginpenalty .int_set:c = { l__enumext_beginparpenalty_#2_int },
927          beginpenalty .initial:n = -51,
928          beginpenalty .value_required:n = true,
929          midpenalty   .int_set:c = { l__enumext_itempenalty_#2_int },
930          midpenalty   .initial:n = -51,
931          midpenalty   .value_required:n = true,
932          endpenalty   .int_set:c = { l__enumext_endparpenalty_#2_int },
933          endpenalty   .initial:n = -51,
934          endpenalty   .value_required:n = true,
935        }
936    }
937  \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

13.17 Setting keys for vertical spaces

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

```
938  \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
939    {
940      \keys_define:nn { enumext / #1 }
941        {
942          topsep    .skip_set:c = { l__enumext_topsep_#2_skip },
943          topsep    .initial:n  = {#3},
944          topsep    .value_required:n = true,
945          partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
946          partopsep .initial:n  = {#4},
947          partopsep .value_required:n = true,
948          parsep    .skip_set:c = { l__enumext_parsep_#2_skip },
949          parsep    .initial:n  = {#5},
950          parsep    .value_required:n = true,
951          itemsep   .skip_set:c = { l__enumext_itemsep_#2_skip },
952          itemsep   .initial:n  = {#6},
953          itemsep   .value_required:n = true,
954          noitemsep .meta:n     = { itemsep = 0pt, parsep = 0pt },
955          noitemsep .value_forbidden:n = true,
956          nosep     .meta:n     = {
957                                  itemsep = 0pt, parsep= 0pt,
958                                  topsep = 0pt, partopsep = 0pt,
959                                  },
960          nosep     .value_forbidden:n = true,
961        }
962    }
```

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

```
963  \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
964    { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
965    { 4.0pt plus 2.0pt minus 1.0pt }
966  \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
967    { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
968    { 2.0pt plus 1.0pt minus 1.0pt }
```

```
969  \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
970    { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
971  \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
972    { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
973  \__enumext_tmp:nnnnnn { keyans  } { v }{ 4.0pt plus 2.0pt minus 1.0pt }
974    { 2.0pt plus 1.0pt minus 1.0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
975    { 2.0pt plus 1.0pt minus 1.0pt }
976  \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
977    { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
978    { 4.0pt plus 2.0pt minus 1.0pt }
979  \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
980    { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
981    { 2.0pt plus 1.0pt minus 1.0pt }
```

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

13.18   Setting `base-fix` key

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

base-fix
__enumext_nested_base_line_fix:

We define the key base-fix only for the *"first level"* of enumext environment.

```
982  \keys_define:nn { enumext /  level-1 }
983    {
984      base-fix .bool_set:N = \l__enumext_base_line_fix_bool,
985      base-fix .initial:n  = false,
986      base-fix .value_forbidden:n = true,
987    }
```

The function __enumext_nested_base_line_fix: passed to the __enumext_parse_keys:n function in the definition of the enumext environment (§13.42) will be responsible for applying the *baseline correction* and adjusting the ⟨*keys*⟩ for the enumext environment and the \printkeyans with *starred argument* '*' (§13.50).

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

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

```
988  \cs_new_protected:Nn \__enumext_nested_base_line_fix:
989    {
990      \bool_lazy_all:nT
991        {
992          { \bool_if_p:N \l__enumext_starred_first_bool }
993          { \bool_if_p:N \l__enumext_base_line_fix_bool }
994          { \bool_not_p:n { \l__enumext_print_keyans_star_bool } }
995        }
996        {
997          \mode_leave_vertical:
998          \vspace { -\dim_eval:n { \baselineskip + \parsep } }
999          \keys_set:nn { enumext / level-1 }
1000            {
1001              topsep = 0pt, above = 0pt, above* = 0pt,
1002            }
1003        }
```

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.

```
1004      \bool_lazy_and:nnT
1005        { \bool_if_p:N \l__enumext_starred_first_bool }
1006        { \bool_if_p:N \l__enumext_print_keyans_star_bool }
1007        {
1008          \mode_leave_vertical:
1009          \skip_vertical:n { -\baselineskip }
1010          \skip_vertical:N \c_zero_skip
1011          \keys_set:nn { enumext / level-1 }
1012            {
1013              topsep = 0pt, above = 0pt, above* = 0pt,
```

```
1014              }
1015          }
1016        \bool_set_false:N \l__enumext_base_line_fix_bool
1017      }
```

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

13.19 Setting keys for horizontal spaces

itemindent
rightmargin
listparindent
list-offset
list-indent

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

```
1018 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1019    {
1020      \keys_define:nn { enumext / #1 }
1021        {
1022          itemindent    .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
1023          itemindent    .value_required:n = true,
1024          rightmargin   .dim_set:c = { l__enumext_rightmargin_#2_dim },
1025          rightmargin   .value_required:n = true,
1026          listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
1027          listparindent .value_required:n = true,
1028          list-offset   .dim_set:c = { l__enumext_listoffset_#2_dim },
1029          list-offset   .value_required:n = true,
1030          list-indent   .code:n    =
1031                          \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
1032                          \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
1033          list-indent   .value_required:n = true,
1034        }
1035    }
1036 \clist_map_inline:nn
1037    {
1038      {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
1039    }
1040    { \__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.

```
1041 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1042    {
1043      \keys_define:nn { enumext / #1 }
1044        {
1045          itemindent    .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
1046          itemindent    .value_required:n = true,
1047          rightmargin   .dim_set:c = { l__enumext_rightmargin_#2_dim },
1048          rightmargin   .value_required:n = true,
1049          listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
1050          listparindent .value_required:n = true,
1051          list-offset   .dim_set:c = { l__enumext_listoffset_#2_dim },
1052          list-offset   .value_required:n = true,
1053          list-indent   .meta:n    = { list-offset = ##1 },
1054          list-indent   .value_required:n = true,
1055        }
1056    }
1057 \clist_map_inline:nn
1058    {
1059      {enumext*}{vii}, {keyans*}{viii}
1060    }
1061    { \__enumext_tmp:nn #1 }
```

13.19.1 Functions for setting the fake itemindent

__enumext_fake_item_indent:
__enumext_keyans_fake_item_indent:
__enumext_fake_item_indent_vii:
__enumext_fake_item_indent_viii:

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.

```
1062 \cs_set_protected:Nn \__enumext_fake_item_indent:
1063    {
1064      \dim_compare:nNnT
1065        { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
1066        >
1067        { \c_zero_dim }
```

```
1068        {
1069            \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
1070              {
1071                \exp_not:N \mode_leave_vertical:
1072                \exp_not:n { \skip_horizontal:n }
1073                  { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
1074                \exp_not:N \ignorespaces
1075              }
1076          }
1077      }
1078  \cs_set_protected:Nn \__enumext_keyans_fake_item_indent:
1079      {
1080        \dim_compare:nNnT
1081          { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
1082          {
1083            \tl_set:Ne \l__enumext_fake_item_indent_v_tl
1084              {
1085                \exp_not:N \mode_leave_vertical:
1086                \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
1087                \exp_not:N \ignorespaces
1088              }
1089          }
1090      }
1091  \cs_set_protected:Nn \__enumext_fake_item_indent_vii:
1092      {
1093        \dim_compare:nNnT
1094          { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
1095          {
1096            \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
1097              {
1098                \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
1099                \exp_not:N \ignorespaces
1100              }
1101          }
1102      }
1103  \cs_set_protected:Nn \__enumext_fake_item_indent_viii:
1104      {
1105        \dim_compare:nNnT
1106          { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
1107          {
1108            \tl_set:Ne \l__enumext_fake_item_indent_viii_tl
1109              {
1110                \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
1111                \exp_not:N \ignorespaces
1112              }
1113          }
1114      }
```

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

13.20   Setting show-length key

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

```
1115  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1116      {
1117        \keys_define:nn { enumext / #1 }
1118          {
1119            show-length .bool_set:c = { l__enumext_show_length_#2_bool },
1120            show-length .initial:n  = false,
1121          }
1122      }
1123  \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

13.21   Setting before, after and first keys

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

```
1124 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1125   {
1126     \keys_define:nn { enumext / #1 }
1127       {
1128         before  .tl_set:c   = { l__enumext_before_no_starred_key_#2_tl },
1129         before  .value_required:n = true,
1130         before* .tl_set:c   = { l__enumext_before_starred_key_#2_tl },
1131         before* .value_required:n = true,
1132         after   .tl_set:c   = { l__enumext_after_stop_list_#2_tl },
1133         after   .value_required:n = true,
1134         first   .tl_set:c   = { l__enumext_after_list_args_#2_tl },
1135         first   .value_required:n = true,
1136       }
1137   }
1138 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

13.21.1 Functions for before, after and first keys in enumext

__enumext_before_args_exec:
__enumext_before_keys_exec:
__enumext_after_stop_list:
__enumext_after_args_exec:

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

```
1139 \cs_new_protected:Nn \__enumext_before_args_exec:
1140   {
1141     \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
1142   }
```

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*⟩}}

```
1143 \cs_new_protected:Nn \__enumext_before_keys_exec:
1144   {
1145     \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
1146   }
```

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

```
1147 \cs_new_protected:Nn \__enumext_after_stop_list:
1148   {
1149     \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
1150   }
```

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.

```
1151 \cs_new_protected:Nn \__enumext_after_args_exec:
1152   {
1153     \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
1154   }
```

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

13.21.2 Functions for before, after and first keys in keyans

__enumext_before_args_exec_v:
__enumext_before_keys_exec_v:
__enumext_after_stop_list_v:
__enumext_after_args_exec_v:

Same implementation as the one used in the enumext environment.

```
1155 \cs_new_protected:Nn \__enumext_before_args_exec_v:
1156   {
1157     \tl_use:N \l__enumext_before_starred_key_v_tl
1158   }
1159 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
1160   {
1161     \tl_use:N \l__enumext_before_no_starred_key_v_tl
1162   }
1163 \cs_new_protected:Nn \__enumext_after_stop_list_v:
1164   {
1165     \tl_use:N \l__enumext_after_stop_list_v_tl
1166   }
1167 \cs_new_protected:Nn \__enumext_after_args_exec_v:
1168   {
1169     \tl_use:N \l__enumext_after_list_args_v_tl
1170   }
```

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

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

`\__enumext_before_args_exec_vii:`
`\__enumext_before_keys_exec_vii`
`\__enumext_after_stop_list_vii:`
`\__enumext_after_args_exec_vii:`

Same implementation as the one used in the enumext environment.

```
1171 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
1172   {
1173     \tl_use:N \l__enumext_before_starred_key_vii_tl
1174   }
1175 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
1176   {
1177     \tl_use:N \l__enumext_before_starred_key_viii_tl
1178   }
1179 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
1180   {
1181     \tl_use:N \l__enumext_before_no_starred_key_vii_tl
1182   }
1183 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
1184   {
1185     \tl_use:N \l__enumext_before_no_starred_key_viii_tl
1186   }
1187 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
1188   {
1189     \tl_use:N \l__enumext_after_stop_list_vii_tl
1190   }
1191 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
1192   {
1193     \tl_use:N \l__enumext_after_stop_list_viii_tl
1194   }
1195 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
1196   {
1197     \tl_use:N \l__enumext_after_list_args_vii_tl
1198   }
1199 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
1200   {
1201     \tl_use:N \l__enumext_after_list_args_viii_tl
1202   }
```

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

13.22 Setting keys for `multicols` and `minipage`

`mini-env`
`mini-sep`
`columns-sep`
`columns`

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.

```
1203 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1204   {
1205     \keys_define:nn { enumext / #1 }
1206       {
1207         mini-env    .dim_set:c  = { l__enumext_minipage_right_#2_dim },
1208         mini-env    .value_required:n = true,
1209         mini-sep    .dim_set:c  = { l__enumext_minipage_hsep_#2_dim },
1210         mini-sep    .initial:n  = 0.3333em,
1211         mini-sep    .value_required:n = true,
1212         columns-sep .dim_set:c  = { l__enumext_columns_sep_#2_dim },
1213         columns-sep .value_required:n = true,
1214         columns     .int_set:c  = { l__enumext_columns_#2_int },
1215         columns     .initial:n  = 1,
1216         columns     .value_required:n = true,
1217       }
1218   }
1219 \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.

```
1220 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1221   {
1222     \keys_define:nn { enumext / #1 }
1223       {
1224         mini-right  .tl_gset:c = { g__enumext_miniright_code_#2_tl },
1225         mini-right  .value_required:n = true,
1226         mini-right* .code:n    = {
```

```
1227                                          \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
1228                                          \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1229                                        },
1230               mini-right* .value_required:n = true,
1231           }
1232     }
1233 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }
```

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

13.23 Adjustment of vertical spaces for 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, this can be seen in 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.

13.23.1 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*.

```
1234 \cs_new_protected:Nn \__enumext_multi_set_vskip:
1235   {
1236     \skip_set:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1237       {
1238         \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1239       }
1240     \skip_set:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1241       {
1242         \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1243       }
1244     \__enumext_add_pre_parsep:
1245   }
```

(*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*.

```
1246 \cs_new_protected:Nn \__enumext_add_pre_parsep:
1247   {
1248     \int_case:nn { \l__enumext_level_int }
1249       {
1250         { 2 }{
1251               \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1252                 {
1253                   \skip_add:Nn \l__enumext_multicols_above_ii_skip
1254                     {
1255                       \l__enumext_parsep_i_skip
1256                     }
1257                 }
1258             }
1259         { 3 }{
```

```
1260                \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1261                  {
1262                    \skip_add:Nn \l__enumext_multicols_above_iii_skip
1263                      {
1264                        \l__enumext_parsep_ii_skip
1265                      }
1266                  }
1267              }
1268          { 4 }{
1269                \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1270                  {
1271                    \skip_add:Nn \l__enumext_multicols_above_iv_skip
1272                      {
1273                        \l__enumext_parsep_iii_skip
1274                      }
1275                  }
1276              }
1277          }
1278      }
```

(*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*⟩.

```
1279  \cs_new_protected:Nn \__enumext_multi_addvspace:
1280    {
1281      \__enumext_multi_set_vskip:
1282      \mode_if_vertical:T
1283        {
1284          \skip_add:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1285            {
1286              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1287            }
1288          \skip_add:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1289            {
1290              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1291            }
1292        }
1293      \par\nopagebreak
1294      \addvspace{ \skip_use:c { l__enumext_multicols_above_ \__enumext_level: _skip } }
1295    }
```

(*End of definition for* __enumext_multi_addvspace:.)

13.23.2 Adjustment of vertical spaces for multicols in keyans

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

```
1296  \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1297    {
1298      \skip_set:Nn \l__enumext_multicols_above_v_skip
1299        {
1300          \l__enumext_topsep_v_skip
1301        }
1302      \skip_set:Nn \l__enumext_multicols_below_v_skip
1303        {
1304          \l__enumext_topsep_v_skip
1305        }
1306    }
1307  \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1308    {
1309      \__enumext_keyans_multi_set_vskip:
1310      \mode_if_vertical:T
1311        {
1312          \skip_add:Nn \l__enumext_multicols_above_v_skip
1313            {
1314              \skip_use:N \l__enumext_partopsep_v_skip
1315            }
1316          \skip_add:Nn \l__enumext_multicols_below_v_skip
1317            {
```

```
1318            \skip_use:N \l__enumext_partopsep_v_skip
1319          }
1320        }
1321      \par\nopagebreak
1322      \addvspace{ \l__enumext_multicols_above_v_skip }
1323    }
```

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

13.24   Adjustment of vertical spaces for `minipage`

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

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

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

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

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

13.24.1   Adjustment of vertical spaces for `minipage` in enumext

__enumext_minipage_set_skip:
__enumext_minipage_add_space:

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

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

```
1324  \cs_new_protected:Nn \__enumext_minipage_set_skip:
1325    {
1326      \skip_set:Nn \l__enumext_minipage_right_skip
1327        {
1328          \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1329        }
1330      \mode_if_vertical:T
1331        {
1332          \skip_add:Nn \l__enumext_minipage_right_skip
1333            {
1334              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1335            }
1336        }
1337      \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:`.

```
1338      \skip_set_eq:cN
1339        { l__enumext_multicols_above_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1340      \skip_set_eq:cN
1341        { l__enumext_multicols_below_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1342      \__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`.

```
1343      \int_compare:nNnT
1344        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1345        {
1346          \skip_zero:N \topskip
1347          \skip_set_eq:Nc \multicolsep { l__enumext_multicols_above_ \__enumext_level: _skip }
```

```
1348            }
1349        }
```

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

```
1350   \cs_new_protected:Nn \__enumext_minipage_add_space:
1351     {
1352        \__enumext_minipage_set_skip:
1353        \__enumext_unskip_unkern:
1354        \mode_if_vertical:TF
1355          {
1356             \nopagebreak\nointerlineskip
1357          }
1358          {
1359             \par\nopagebreak\nointerlineskip
1360             \skip_zero:c { l__enumext_partopsep_ \__enumext_level: _skip }
1361          }
1362        \int_compare:nNnTF
1363          { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1364          {
1365             \addvspace{ 0.445\box_ht:N \strutbox }
1366          }
1367          {
1368             \addvspace{ 0.250\box_ht:N \strutbox }
1369          }
1370     }
```

(*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.

```
1371   \cs_new_protected:Nn \__enumext_pre_itemsep_skip:
1372     {
1373        \int_case:nn { \l__enumext_level_int }
1374          {
1375             { 2 }{
1376                \skip_if_eq:nnTF
1377                  { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1378                  {
1379                     \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1380                     \skip_set:Nn \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1381                  }
1382                  {
1383                     \dim_compare:nNnT
1384                       { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1385                       {
1386                          \skip_sub:Nn
1387                            \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1388                          \skip_sub:Nn
1389                            \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1390                          \skip_add:Nn
1391                            \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1392                          \skip_add:Nn
1393                            \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1394                       }
1395                     \dim_compare:nNnT
1396                       { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1397                       {
1398                          \skip_set:Nn \l__enumext_minipage_temp_skip
1399                            {
1400                               \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1401                            }
1402                          \skip_sub:Nn
1403                            \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1404                          \skip_sub:Nn
1405                            \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1406                          \skip_add:Nn
```

```
1407                        \l__enumext_minipage_after_skip
1408                          { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1409                        \skip_add:Nn
1410                        \l__enumext_multicols_below_ii_skip
1411                          { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1412                      }
1413                  }
1414              }
1415          { 3 }{
1416              \skip_if_eq:nnTF
1417                { \l__enumext_itemsep_ii_skip } { \c_zero_skip }
1418                {
1419                  \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1420                  \skip_set:Nn \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1421                }
1422                {
1423                  \dim_compare:nNnT
1424                    { \l__enumext_itemsep_ii_skip } < { \l__enumext_minipage_after_skip }
1425                    {
1426                      \skip_sub:Nn
1427                        \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1428                      \skip_sub:Nn
1429                        \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1430                      \skip_add:Nn
1431                        \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1432                      \skip_add:Nn
1433                        \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1434                    }
1435                  \dim_compare:nNnT
1436                    { \l__enumext_itemsep_ii_skip } > { \l__enumext_minipage_after_skip }
1437                    {
1438                      \skip_set:Nn \l__enumext_minipage_temp_skip
1439                        {
1440                          \l__enumext_itemsep_ii_skip - \l__enumext_minipage_after_skip
1441                        }
1442                      \skip_sub:Nn
1443                        \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1444                      \skip_sub:Nn
1445                        \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1446                      \skip_add:Nn
1447                        \l__enumext_minipage_after_skip
1448                          { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1449                      \skip_add:Nn
1450                        \l__enumext_multicols_below_iii_skip
1451                          { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1452                    }
1453                }
1454            }
1455          { 4 }{
1456              \skip_if_eq:nnTF { \l__enumext_itemsep_iii_skip } { \c_zero_skip }
1457                {
1458                  \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1459                  \skip_set:Nn \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1460                }
1461                {
1462                  \dim_compare:nNnT
1463                    { \l__enumext_itemsep_iii_skip } < { \l__enumext_minipage_after_skip }
1464                    {
1465                      \skip_sub:Nn
1466                        \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1467                      \skip_sub:Nn
1468                        \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1469                      \skip_add:Nn
1470                        \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1471                      \skip_add:Nn
1472                        \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1473                    }
1474                  \dim_compare:nNnT
1475                    { \l__enumext_itemsep_iii_skip } > { \l__enumext_minipage_after_skip }
1476                    {
1477                      \skip_set:Nn \l__enumext_minipage_temp_skip
```

```
1478                                 {
1479                                     \l__enumext_itemsep_iii_skip - \l__enumext_minipage_after_skip
1480                                 }
1481                             \skip_sub:Nn
1482                                 \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1483                             \skip_sub:Nn
1484                                 \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1485                             \skip_add:Nn
1486                                 \l__enumext_minipage_after_skip
1487                                 { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1488                             \skip_add:Nn
1489                                 \l__enumext_multicols_below_iv_skip
1490                                 { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1491                         }
1492                     }
1493                 }
1494             }
1495     }
```

(*End of definition for* `\__enumext_pre_itemsep_skip:`.)

13.24.2 Adjustment of vertical spaces for `minipage` in `keyans`

`\__enumext_keyans_minipage_set_skip:`
`\__enumext_keyans_minipage_add_space:`
`\__enumext_keyans_pre_itemsep_skip:`

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

```
1496 \cs_new_protected:Nn \__enumext_keyans_minipage_set_skip:
1497     {
1498         \skip_zero:N \l__enumext_minipage_after_skip
1499         \skip_zero:N \l__enumext_minipage_left_skip
1500         \skip_zero:N \l__enumext_minipage_right_skip
1501         \skip_set:Nn \l__enumext_minipage_right_skip
1502             {
1503                 \l__enumext_topsep_v_skip
1504             }
1505         \mode_if_vertical:T
1506             {
1507                 \skip_add:Nn \l__enumext_minipage_right_skip
1508                     {
1509                         \l__enumext_partopsep_v_skip
1510                     }
1511             }
1512         \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip
1513         \skip_set_eq:NN \l__enumext_multicols_above_v_skip \l__enumext_minipage_right_skip
1514         \skip_set_eq:NN \l__enumext_multicols_below_v_skip \l__enumext_minipage_right_skip
1515         \__enumext_keyans_pre_itemsep_skip:
1516         \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
1517             {
1518                 \skip_zero:N \topskip
1519                 \skip_set_eq:NN \multicolsep \l__enumext_minipage_right_skip
1520             }
1521     }
1522 \cs_new_protected:Nn \__enumext_keyans_minipage_add_space:
1523     {
1524         \__enumext_keyans_minipage_set_skip:
1525         \__enumext_unskip_unkern:
1526         \mode_if_vertical:TF
1527             {
1528                 \nopagebreak\nointerlineskip
1529             }
1530             {
1531                 \par\nopagebreak\nointerlineskip
1532                 \skip_zero:N \l__enumext_partopsep_v_skip
1533             }
1534         \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1535             {
1536                 \addvspace{ 0.445\box_ht:N \strutbox }
1537             }
1538             {
1539                 \addvspace{ 0.250\box_ht:N \strutbox }
1540             }
1541     }
```

```
1542 \cs_new_protected:Nn \__enumext_keyans_pre_itemsep_skip:
1543   {
1544     \skip_if_eq:nnTF
1545       { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1546       {
1547         \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1548         \skip_set:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1549       }
1550       {
1551         \dim_compare:nNnT
1552           { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1553           {
1554             \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1555             \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1556             \skip_add:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1557             \skip_add:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1558           }
1559         \dim_compare:nNnT
1560           { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1561           {
1562             \skip_set:Nn \l__enumext_minipage_temp_skip
1563               {
1564                 \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1565               }
1566             \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1567             \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1568             \skip_add:Nn \l__enumext_minipage_after_skip
1569               { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1570             \skip_add:Nn \l__enumext_multicols_below_v_skip
1571               { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1572           }
1573       }
1574   }
```

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

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

__enumext_mini_set_vskip_vii:
__enumext_mini_set_vskip_viii:

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

```
1575 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1576   {
1577     \skip_zero_new:N \l__enumext_minipage_left_skip
1578     \skip_gzero_new:N \g__enumext_minipage_right_skip
1579     \skip_gzero_new:N \g__enumext_minipage_after_skip
1580     \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1581       {
1582         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1583         \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1584       }
1585       {
1586         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1587         \skip_gset:Nn \g__enumext_minipage_right_skip
1588           {
1589             \l__enumext_topsep_vii_skip
1590           }
1591         \skip_gset:Nn \g__enumext_minipage_after_skip
1592           {
1593             0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1594           }
1595       }
1596   }
1597 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1598   {
1599     \skip_zero_new:N \l__enumext_minipage_after_skip
1600     \skip_zero_new:N \l__enumext_minipage_left_skip
1601     \skip_zero_new:N \l__enumext_minipage_right_skip
1602     \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1603       {
1604         \skip_set:Nn \l__enumext_minipage_left_skip
```

```
1605                    {
1606                        0.5\box_dp:N \strutbox
1607                    }
1608                \skip_set:Nn \l__enumext_minipage_right_skip
1609                    {
1610                        \l__enumext_partopsep_viii_skip
1611                    }
1612                \skip_set:Nn \l__enumext_minipage_after_skip
1613                    {
1614                        1.6\box_dp:N \strutbox
1615                    }
1616            }
1617            {
1618                \skip_set:Nn \l__enumext_minipage_left_skip
1619                    {
1620                        0.5875\box_dp:N \strutbox
1621                    }
1622                \skip_set:Nn \l__enumext_minipage_right_skip
1623                    {
1624                        \l__enumext_topsep_viii_skip
1625                    }
1626                \skip_set:Nn \l__enumext_minipage_after_skip
1627                    {
1628                        0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1629                    }
1630            }
1631    }
```

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

```
1632    \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1633    {
1634        \__enumext_mini_set_vskip_vii:
1635        \par\nopagebreak
1636        \addvspace { \l__enumext_minipage_left_skip }
1637    }
1638    \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1639    {
1640        \__enumext_mini_set_vskip_viii:
1641        \par\nopagebreak
1642        \addvspace { \l__enumext_minipage_left_skip }
1643    }
```

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

13.24.4   The command \miniright

The command \miniright will close the __enumext_mini_page environment on the *"left side"*, open the
__enumext_mini_page environment on the *"right side"* adding the *adjusted vertical space*. By default we will
add \centering when starting the *"right side"* environment. The *starred argument* '*' inhibits the use of
\centering command i.e. the usual 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 environ-
ment or somewhere inappropriate then we will call the internal functions to execute it in the enumext and
keyans environments.

```
1644    \NewDocumentCommand \miniright { s }
1645    {
1646        \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1647            {
1648                \msg_error:nnn { enumext } { wrong-miniright-place }
1649            }
1650        % outside
1651        \bool_lazy_and:nnT
1652            { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
1653            { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
```

```
1654        {
1655            \msg_error:nnn { enumext } { wrong-miniright-place }
1656        }
1657    % starred env
1658    \bool_lazy_and:nnT
1659        { \bool_if_p:N \g__enumext_starred_bool }
1660        { \bool_not_p:n { \l__enumext_standar_bool } }
1661        {
1662            \msg_error:nnn { enumext } { wrong-miniright-starred }
1663        }
1664    % exec
1665    \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
1666        {
1667            \__enumext_keyans_mini_right_cmd:n {#1}
1668        }
1669        { \__enumext_mini_right_cmd:n {#1} }
1670    }
```

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

`\__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.

```
1671    \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1672    {
1673        \dim_compare:nNnTF
1674            { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1675            {
1676                \__enumext_multicols_stop:
1677                \int_compare:nNnT
1678                    { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } = { 1 }
1679                    {
1680                        \par\addvspace{ \l__enumext_minipage_after_skip }
1681                    }
1682                \end__enumext_mini_page
1683                \hfill
1684                \__enumext_mini_page{ \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } }
1685                    \par\nointerlineskip
1686                    \addvspace { \l__enumext_minipage_right_skip }
1687                    \bool_if:nF {#1}
1688                        {
1689                            \centering
1690                        }
1691                    \int_gzero:N \g__enumext_minipage_stat_int
1692                }
1693            { \msg_error:nnn { enumext } { wrong-miniright-use } }
1694        % paranoia
1695        \RenewDocumentCommand \miniright { s }
1696            {
1697                \msg_error:nn { enumext } { many-miniright-used }
1698            }
1699    }
```

(*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.

```
1700    \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1701    {
1702        \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1703            {
1704                \__enumext_keyans_multicols_stop:
1705                \int_compare:nNnT { \l__enumext_columns_v_int } = { 1 }
1706                    {
1707                        \par\addvspace{ \l__enumext_minipage_after_skip }
1708                    }
1709                \end__enumext_mini_page
```

```
1710        \hfill
1711        \__enumext_mini_page{ \l__enumext_minipage_right_v_dim }
1712          \par\nointerlineskip
1713          \addvspace { \l__enumext_minipage_right_skip }
1714          \bool_if:nF {#1}
1715            {
1716               \centering
1717            }
1718          \int_gzero:N \g__enumext_minipage_stat_int
1719        }
1720      { \msg_error:nnn { enumext } { wrong-miniright-use } }
1721    % paranoia
1722    \RenewDocumentCommand \miniright { s }
1723      {
1724        \msg_error:nn { enumext } { many-miniright-used }
1725      }
1726  }
```

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

13.25   Setting above and below keys

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

above
above*
below
below*

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

```
1727  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1728    {
1729      \keys_define:nn { enumext / #1 }
1730        {
1731          above  .skip_set:c = { l__enumext_vspace_above_#2_skip },
1732          above  .value_required:n = true,
1733          above* .code:n      = \bool_set_true:c { l__enumext_vspace_a_star_#2_bool }
1734                                 \keys_set:nn { enumext / #1 } { above = {##1} },
1735          above* .value_required:n = true,
1736          below  .skip_set:c = { l__enumext_vspace_below_#2_skip },
1737          below  .value_required:n = true,
1738          below* .code:n      = \bool_set_true:c { l__enumext_vspace_b_star_#2_bool }
1739                                 \keys_set:nn { enumext / #1 } { below = {##1} },
1740          below* .value_required:n = true,
1741        }
1742    }
1743  \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

13.25.1   Functions for above and below keys in enumext

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

```
1744  \cs_new_protected:Nn \__enumext_vspace_above:
1745    {
1746      \skip_if_eq:nnF
1747        { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1748        {
1749          \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
1750            {
1751              \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1752            }
1753            {
1754              \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1755            }
1756        }
1757    }
```

(*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.

```
1758 \cs_new_protected:Nn \__enumext_vspace_below:
1759   {
1760     \skip_if_eq:nnF
1761       { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1762       {
1763         \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1764           {
1765             \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1766           }
1767           {
1768             \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1769           }
1770       }
1771   }
```

(*End of definition for* __enumext_vspace_below:.)

13.25.2 Functions for above and below keys in keyans

__enumext_vspace_above_v:

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

```
1772 \cs_new_protected:Nn \__enumext_vspace_above_v:
1773   {
1774     \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1775       {
1776         \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1777           {
1778             \vspace*{ \l__enumext_vspace_above_v_skip }
1779           }
1780           { \vspace { \l__enumext_vspace_above_v_skip } }
1781       }
1782   }
```

(*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.

```
1783 \cs_new_protected:Nn \__enumext_vspace_below_v:
1784   {
1785     \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1786       {
1787         \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1788           {
1789             \vspace*{ \l__enumext_vspace_below_v_skip }
1790           }
1791           { \vspace { \l__enumext_vspace_below_v_skip } }
1792       }
1793   }
```

(*End of definition for* __enumext_vspace_below_v:.)

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

__enumext_vspace_above_vii:
__enumext_vspace_above_viii:

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.

```
1794 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1795   {
1796     \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1797       {
1798         \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1799           {
1800             \vspace*{ \l__enumext_vspace_above_vii_skip }
1801           }
1802           { \vspace { \l__enumext_vspace_above_vii_skip } }
1803       }
1804   }
1805 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1806   {
1807     \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1808       {
```

```
1809          \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1810            {
1811              \vspace*{ \l__enumext_vspace_above_viii_skip }
1812            }
1813            { \vspace { \l__enumext_vspace_above_viii_skip } }
1814          }
1815       }
```

(*End of definition for* __enumext_vspace_above_vii: *and* __enumext_vspace_above_viii:.)

__enumext_vspace_below_vii:
__enumext_vspace_below_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.

```
1816 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1817    {
1818      \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1819        {
1820          \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1821            {
1822              \vspace*{ \l__enumext_vspace_below_vii_skip }
1823            }
1824            { \vspace { \l__enumext_vspace_below_vii_skip } }
1825        }
1826    }
1827 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1828    {
1829      \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1830        {
1831          \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1832            {
1833              \vspace*{ \l__enumext_vspace_below_viii_skip }
1834            }
1835            { \vspace { \l__enumext_vspace_below_viii_skip } }
1836        }
1837    }
```

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

13.26   Setting series, resume and resume* keys

The series key is responsible for the whole process of the resume and resume* keys. The idea behind this is to be able to absorb the ⟨keys⟩ passed to the *optional argument* of the environments enumext and enumext*, but, discarding some specific ⟨keys⟩.

series
resume
resume*

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

In version 1.6 it is allowed to pass the key resume *without value* by means of the command \setenumext, for the correct operation of this we must set the boolean variable \l__enumext_resume_count_bool set by the key resume *without value* to "true" to be later processed by the function __enumext_parse_series:n in the definition of the environments enumext and enumext*.

```
1838 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1839    {
1840      \keys_define:nn { enumext / #1 }
1841        {
1842          series  .str_set:N = \l__enumext_series_name_str,
1843          series  .value_required:n = true,
1844          resume  .code:n = {
1845                          \bool_if:NF \l__enumext_print_keyans_cmd_bool
1846                            {
1847                              \tl_set:Nn \l__enumext_series_name_tl {##1}
1848                              \tl_if_empty:NTF \l__enumext_series_name_tl
1849                                {
1850                                  \bool_set_true:c { l__enumext_resume_count_#2_bool }
1851                                  \bool_set_eq:Nc
1852                                    \l__enumext_resume_count_bool
1853                                    { l__enumext_resume_count_#2_bool }
1854                                }
1855                                {
1856                                  \__enumext_resume:n {##1}
1857                                }
1858                                  }
```

```
1859                            },
1860              resume* .code:n = {
1861                              \bool_if:NF \l__enumext_print_keyans_cmd_bool
1862                                {
1863                                  \bool_set_true:c { l__enumext_resume_count_#2_bool }
1864                                  \bool_set_eq:Nc
1865                                    \l__enumext_resume_count_bool { l__enumext_resume_count_#2_bool
1866                                  \bool_set_true:c { l__enumext_resume_star_key_#2_bool }
1867                                  \__enumext_resume_star:
1868                                }
1869                            },
1870              resume* .value_forbidden:n = true,
1871        }
1872    }
1873  \clist_map_inline:nn
1874    {
1875      {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},{enumext*}{vii},
1876    }
1877    { \__enumext_tmp:nn #1 }
```

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

13.26.1   Internal function to save counter and integer values

__enumext_standar_save_counter:
__enumext_standar_save_counter_aux:
__enumext_starred_save_counter:
__enumext_starred_save_counter_aux:

The __enumext_standar_save_counter: and __enumext_starred_save_counter: functions will save the last counter value to \g__enumext_series_⟨*series name*⟩_int if the series={⟨*series name*⟩} key has been passed, to \c@__enumext_resume_X_int if it has passed the key resume *without value* and the key series is not active, in \g__enumext_series_⟨*series name*⟩_X_int if the key resume={⟨*series name*⟩} has been passed and in \g__enumext_series_⟨*store name*⟩_X_int if the key has been passed save-ans={⟨*store name*⟩}.

🍐 The variables \l__enumext_series_name_str and \l__enumext_series_name_tl contain the same {⟨*series name*⟩} but are executed at different moments, the integer variable with \l__enumext_series_name_str sets the value when execute series={⟨*series name*⟩} and the integer variable with \l__enumext_series_name_tl sets the subsequent values when use resume={⟨*series name*⟩}. This function is passed to the enumext environment definition (§13.42) and the enumext* environment definition (§13.47).

```
1878  \cs_new_protected:Nn \__enumext_standar_save_counter:
1879    {
1880      \bool_if:NTF \g__enumext_standar_bool
1881        {
1882          \__enumext_standar_save_counter_aux:
1883          \int_compare:nNnT { \l__enumext_level_int } = { 1 }
1884            {
1885              \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1886                {
1887                  \int_gset_eq:cN
1888                    { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1889                }
1890            }
1891        }
1892        {
1893          \__enumext_standar_save_counter_aux:
1894        }
1895    }
1896  \cs_new_protected:Nn \__enumext_standar_save_counter_aux:
1897    {
1898      \str_if_empty:NF \l__enumext_series_name_str
1899        {
1900          \int_gset_eq:cc
1901            { g__enumext_series_ \l__enumext_series_name_str _ \__enumext_level: _int }
1902            { c@enumX \__enumext_level: }
1903        }
1904      \tl_if_empty:NTF \l__enumext_series_name_tl
1905        {
1906          \str_if_empty:NT \l__enumext_series_name_str
1907            {
1908              \tl_if_empty:NT \l__enumext_store_name_tl
1909                {
1910                  \int_gset_eq:cc
1911                    { c@ __enumext_resume_ \__enumext_level: _int } { c@enumX \__enumext_level: }
1912                }
1913            }
```

```
1914        }
1915        {
1916            \int_if_exist:cT
1917                { g__enumext_series_ \l__enumext_series_name_tl _ \__enumext_level: _int }
1918                {
1919                    \int_gset_eq:cc
1920                        { g__enumext_series_ \l__enumext_series_name_tl _ \__enumext_level: _int }
1921                        { c@enumX \__enumext_level: }
1922                }
1923        }
1924    }
1925 \cs_new_protected:Nn \__enumext_starred_save_counter:
1926    {
1927        \bool_if:NTF \g__enumext_starred_bool
1928            {
1929                \__enumext_starred_save_counter_aux:
1930                \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1931                    {
1932                        \int_gset_eq:cN
1933                            { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1934                    }
1935            }
1936            {
1937                \__enumext_starred_save_counter_aux:
1938            }
1939    }
1940 \cs_new_protected:Nn \__enumext_starred_save_counter_aux:
1941    {
1942        \str_if_empty:NF \l__enumext_series_name_str
1943            {
1944                \int_gset_eq:cN
1945                    { g__enumext_series_ \l__enumext_series_name_str _vii_int } \value{enumXvii}
1946            }
1947        \tl_if_empty:NTF \l__enumext_series_name_tl
1948            {
1949                \str_if_empty:NT \l__enumext_series_name_str
1950                    {
1951                        \tl_if_empty:NT \l__enumext_store_name_tl
1952                            {
1953                                \int_gset_eq:cc { c@ __enumext_resume_vii_int } { c@enumXvii }
1954                            }
1955                    }
1956            }
1957            {
1958                \int_if_exist:cT { g__enumext_series_ \l__enumext_series_name_tl _vii_int }
1959                    {
1960                        \int_gset_eq:cN
1961                            { g__enumext_series_ \l__enumext_series_name_tl _vii_int } \value{enumXvii}
1962                    }
1963            }
1964    }
```

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

13.26.2 Internal function for resume counters

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

```
1965 \cs_new_protected:Nn \__enumext_resume_counter:
1966    {
1967        \cs_set:Npn \__enumext_tmp:n ##1
1968            {
1969                \bool_if:cT { l__enumext_resume_count_ \int_to_roman:n {##1} _bool }
1970                    {
1971                        \exp_args:Ne \int_set:cn { l__enumext_start_ \int_to_roman:n {##1} _int }
1972                            {
1973                                \int_use:c { c@ __enumext_resume_ \int_to_roman:n {##1} _int } + 1
1974                            }
1975                    }
1976            }
1977        \int_compare:nNnT { \l__enumext_level_int } > { 0 }
```

```
1978            {
1979                \bool_lazy_and:nnTF
1980                    { \bool_if_p:N \l__enumext_standar_first_bool }
1981                    { \bool_if_p:N \l__enumext_store_active_bool }
1982                    {
1983                      \int_set:Nn \l__enumext_start_i_int
1984                          {
1985                            \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1986                          }
1987                    }
1988                    {
1989                      \int_step_function:nN { \l__enumext_level_int } \__enumext_tmp:n
1990                    }
1991              }
1992          \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
1993            {
1994                \bool_lazy_and:nnTF
1995                    { \bool_if_p:N \l__enumext_starred_first_bool }
1996                    { \bool_if_p:N \l__enumext_store_active_bool }
1997                    {
1998                      \int_set:Nn \l__enumext_start_vii_int
1999                          {
2000                            \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
2001                          }
2002                    }
2003                    {
2004                      \bool_if:NT \l__enumext_resume_count_vii_bool
2005                          {
2006                            \int_set:Nn \l__enumext_start_vii_int
2007                              {
2008                                \int_use:c { c@ __enumext_resume_vii_int } + 1
2009                              }
2010                          }
2011                    }
2012              }
2013          }
```

(*End of definition for* `\__enumext_resume_counter:`.)

13.26.3   Internal functions for series key

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

The function `\__enumext_filter_series:n` will be in charge of filtering the ⟨*keys*⟩ we want to store where #1 represents the *optional argument* passed to the environment. This implementation is adapted directly from the code provided by Jonathan P. Spratte (@Skillmon) in chat-TeX-SX

```
2014  \cs_new:Npn \__enumext_filter_series:n #1
2015    {
2016      \use:e
2017        {
2018          \keyval_parse:NNn
2019            \__enumext_filter_series_key:n
2020            \__enumext_filter_series_pair:nn {#1}
2021        }
2022    }
```

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

```
2023  \cs_new:Npn \__enumext_filter_series_key:n #1
2024    {
2025      \str_case:nnF {#1}
2026        {
2027          { resume } {} { resume* } {} { reset } {} { reset* } {} { base-fix } {}
2028        }
2029        { , { \exp_not:n {#1} } }
2030    }
```

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

```
2031  \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
2032    {
2033      \str_case:nnF {#1}
2034        {
2035          { series } {} { resume } {} { start } {}
2036          { start* } {}  { save-ans } {} { save-key } {}
```

```
2037              }
2038              { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2039          }
```

(*End of definition for* __enumext_filter_series:n, __enumext_filter_series_key:n, *and* __enumext_filter_series_-
pair:nn.)

__enumext_save_last_keys:n
__enumext_resume_last_counter:

The function __enumext_save_last_keys:n will be in charge of saving the filtering ⟨*keys*⟩ when the
keys series={⟨*series name*⟩} or resume={⟨*series name*⟩} or resume* are NOT active and will save them
in the variable \g__enumext_resume_last_keys_X_tl for the enumext environment and in the variable
\g__enumext_resume_last_keys_vii_tl for the enumext* environment.

🌀 The boolean variable \l__enumext_resume_series_X_bool is set to "true" by the key resume={⟨*series name*⟩}, the
boolean variable \l__enumext_resume_star_key_X_bool is set to "true" by the key resume*, in this case we need to
make sure both variables are set to ""false" so that they don't override the default filtered ⟨*keys*⟩.

```
2040  \cs_new_protected:Npn \__enumext_save_last_keys:n #1
2041    {
2042      \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2043        {
2044          \bool_if:cF { l__enumext_resume_series_ \__enumext_level: _bool }
2045            {
2046              \bool_if:cF { l__enumext_resume_star_key_ \__enumext_level: _bool }
2047                {
2048                  \tl_gclear:c { g__enumext_resume_last_keys_ \__enumext_level: _tl }
2049                  \tl_gset:ce
2050                    { g__enumext_resume_last_keys_ \__enumext_level: _tl }
2051                    { \__enumext_filter_series:n {#1} }
2052                }
2053            }
2054        }
2055      \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
2056        {
2057          \bool_if:NF \l__enumext_resume_series_vii_bool
2058            {
2059              \tl_gclear:N \g__enumext_resume_last_keys_vii_tl
2060              \tl_gset:Ne \g__enumext_resume_last_keys_vii_tl { \__enumext_filter_series:n {#1} }
2061            }
2062        }
2063    }
```

The __enumext_resume_last_counter: function will be in charge of setting the *"counters"* when the
keys series={⟨*series name*⟩} or resume={⟨*series name*⟩} are NOT active and the resume key is being used
without value either in the *optional argument* of the environments or through the \setenumext command.

🌀 The boolean variable \l__enumext_resume_count_bool is set to "true" by the keys resume *without value* and resume*;
and set to "false" by the keys start and start* (§13.15).

```
2064  \cs_new_protected:Nn \__enumext_resume_last_counter:
2065    {
2066      \bool_lazy_and:nnT
2067        { \bool_if_p:N \l__enumext_resume_count_bool }
2068        { \tl_if_empty_p:N \l__enumext_series_name_tl }
2069        {
2070          \__enumext_resume_counter:
2071        }
2072    }
```

(*End of definition for* __enumext_save_last_keys:n *and* __enumext_resume_last_counter:.)

__enumext_parse_series:n

The __enumext_parse_series:n function handled by the series key will be responsible for *storing* the
filtered ⟨*keys*⟩ from the *optional arguments* of the enumext and enumext* environments for the resume and
resume* keys. If the series key is NOT active it will call the __enumext_save_last_keys:n function
to *store* the *filtered* ⟨*keys*⟩ that will be used by the resume* key and then the __enumext_resume_last_-
counter: function used by the resume key *without value* if it is active, otherwise *store* the *filtered* ⟨*keys*⟩ in
the global variable \g__enumext_series_⟨*series name*⟩_X_tl along with the creation of the integer variable
\g__enumext_series_⟨*series name*⟩_X_int used by the resume key *with value.*

🌀 This function is passed to the function __enumext_parse_keys:n in the enumext environment definition (§13.42) and
to the function __enumext_parse_keys_vii:n in the enumext* environment definition (§13.47).

```
2073  \cs_new_protected:Npn \__enumext_parse_series:n #1
2074    {
2075      \str_if_empty:NTF \l__enumext_series_name_str
2076        {
```

```
2077            \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2078              {
2079                \__enumext_save_last_keys:n {#1}
2080                \__enumext_resume_last_counter:
2081              }
2082            \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
2083              {
2084                \__enumext_save_last_keys:n {#1}
2085                \__enumext_resume_last_counter:
2086              }
2087          }
2088          {
2089            \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2090              {
2091                \tl_gclear_new:c
2092                  { g__enumext_series_ \l__enumext_series_name_str _ \__enumext_level: _tl }
2093                \tl_gset:ce
2094                  { g__enumext_series_ \l__enumext_series_name_str _ \__enumext_level: _tl }
2095                  { \__enumext_filter_series:n {#1} }
2096                \int_if_exist:cF
2097                  { g__enumext_series_ \l__enumext_series_name_str _ \__enumext_level: _int }
2098                  {
2099                    \int_new:c
2100                      { g__enumext_series_ \l__enumext_series_name_str _ \__enumext_level: _int }
2101                  }
2102              }
2103            \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
2104              {
2105                \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_name_str _vii_tl }
2106                \tl_gset:ce
2107                  { g__enumext_series_ \l__enumext_series_name_str _vii_tl }
2108                  { \__enumext_filter_series:n {#1} }
2109                \int_if_exist:cF { g__enumext_series_ \l__enumext_series_name_str _vii_int }
2110                  {
2111                    \int_new:c { g__enumext_series_ \l__enumext_series_name_str _vii_int }
2112                  }
2113              }
2114          }
2115      }
```

(*End of definition for* __enumext_parse_series:n.)

13.26.4　Internal functions for resume key with value

__enumext_resume:n　The function __enumext_resume:n will handle the *argument* {⟨*series name*⟩} passed to the resume key in enumext and enumext* environments. First we will check if the global variable \g__enumext_series_-⟨*series name*⟩_X_tl exists, if so we will call the function __enumext_resume_series:n and pass the ⟨*keys*⟩ stored in \g__enumext_series_⟨*series name*⟩_X_tl to the environments, otherwise we will return an error.

```
2116  \cs_new_protected:Npn \__enumext_resume:n #1
2117    {
2118      \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2119        {
2120          \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _ \__enumext_level: _tl }
2121            {
2122              \__enumext_resume_series:n {#1}
2123              \exp_args:Ne \keys_set:nv { enumext / level-\int_use:N \l__enumext_level_int }
2124                { g__enumext_series_ \tl_to_str:n {#1} _ \__enumext_level: _tl }
2125            }
2126            {
2127              \msg_error:nnn { enumext } { unknown-series-standar } {#1}
2128            }
2129        }
2130      \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
2131        {
2132          \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _vii_tl }
2133            {
2134              \__enumext_resume_series:n {#1}
2135              \keys_set:nv { enumext / enumext* }
2136                { g__enumext_series_ \tl_to_str:n {#1} _vii_tl }
2137            }
2138            {
2139              \msg_error:nnn { enumext } { unknown-series-starred } {#1}
```

```
2140                      }
2141                  }
2142              }
```

(*End of definition for* __enumext_resume:n.)

__enumext_resume_series:n
__enumext_resume_integer_series:

The function __enumext_resume_series:n will set the variable \l__enumext_resume_series_X_bool to "true" and pass the {⟨*argument*⟩} to the variable \l__enumext_series_name_tl then call the function __enumext_resume_integer_series:.

```
2143  \cs_new_protected:Npn \__enumext_resume_series:n #1
2144    {
2145      \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2146        {
2147          \bool_set_true:c { l__enumext_resume_series_ \__enumext_level: _bool }
2148          \tl_clear:N \l__enumext_series_name_tl
2149          \tl_set:Nn \l__enumext_series_name_tl {#1}
2150          \__enumext_resume_integer_series:
2151        }
2152      \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
2153        {
2154          \bool_set_true:N \l__enumext_resume_series_vii_bool
2155          \tl_clear:N \l__enumext_series_name_tl
2156          \tl_set:Nn \l__enumext_series_name_tl {#1}
2157          \__enumext_resume_integer_series:
2158        }
2159    }
```

The function __enumext_resume_integer_series: will be executed when the resume={⟨*series name*⟩} key is active, setting the *start value* for the *"counter"* of the *"current level"* of the environments in which it is run according to the value of the *"integer variables"* created by the series key. If the save-ans key is active it will set the *start value* for the *"counter"* according to the value of the integer variable created by that key.

```
2160  \cs_new_protected:Nn \__enumext_resume_integer_series:
2161    {
2162      \cs_set:Npn \__enumext_tmp:n ##1
2163        {
2164          \int_if_exist:cT { g__enumext_series_ \l__enumext_series_name_tl _ \int_to_roman:n {##1} _
2165            {
2166              \exp_args:Ne \int_set:cn { l__enumext_start_ \int_to_roman:n {##1} _int }
2167                {
2168                  \int_use:c { g__enumext_series_ \l__enumext_series_name_tl _ \int_to_roman:n {##1}
2169                }
2170            }
2171        }
2172      \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2173        {
2174          \bool_lazy_and:nnTF
2175            { \bool_if_p:N \l__enumext_standar_first_bool }
2176            { \bool_if_p:N \l__enumext_store_active_bool }
2177            {
2178              \int_set:Nn \l__enumext_start_i_int
2179                {
2180                  \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
2181                }
2182            }
2183            {
2184              \int_step_function:nN { \l__enumext_level_int } \__enumext_tmp:n
2185            }
2186        }
2187      \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
2188        {
2189          \bool_lazy_and:nnTF
2190            { \bool_if_p:N \l__enumext_starred_first_bool }
2191            { \bool_if_p:N \l__enumext_store_active_bool }
2192            {
2193              \int_set:Nn \l__enumext_start_vii_int
2194                {
2195                  \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
2196                }
2197            }
2198            {
2199              \int_set:Nn \l__enumext_start_vii_int
```

```
2200                    {
2201                        \int_use:c { g__enumext_series_ \l__enumext_series_name_tl _vii_int } + 1
2202                    }
2203                }
2204            }
2205        }
```

(*End of definition for* __enumext_resume_series:n *and* __enumext_resume_integer_series:.)

13.26.5   Internal function for resume* key

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

```
2206  \cs_new_protected:Nn \__enumext_resume_star:
2207    {
2208        \cs_set:Npn \__enumext_tmp:n ##1
2209          {
2210              \tl_if_empty:cTF { g__enumext_resume_last_keys_ \int_to_roman:n {##1} _tl }
2211                {
2212                    \__enumext_resume_counter:
2213                }
2214                {
2215                    \__enumext_resume_counter:
2216                    \exp_args:Ne \keys_set:nv
2217                      { enumext / level-\int_use:N \l__enumext_level_int }
2218                      { g__enumext_resume_last_keys_ \int_to_roman:n {##1} _tl }
2219                }
2220          }
2221        \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2222          {
2223              \int_step_function:nN { \l__enumext_level_int } \__enumext_tmp:n
2224          }
2225        \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
2226          {
2227              \tl_if_empty:NTF \g__enumext_resume_last_keys_vii_tl
2228                {
2229                    \__enumext_resume_counter:
2230                }
2231                {
2232                    \__enumext_resume_counter:
2233                    \keys_set:nV { enumext / enumext* } \g__enumext_resume_last_keys_vii_tl
2234                }
2235          }
2236    }
```

(*End of definition for* __enumext_resume_star:.)

13.27   The \resetenumext command

Sometimes it is necessary to be able to reset the *"counters"* of the environments according to some value, for example \chapter. Since we use *"internal counters"* for the resume and resume* keys which set the *start value*, but are not accessible by the user, it is to provide a public command for this. This implementation is an adaptation of the answers given by Clea F. Rees (@cfr) and Jonathan P. Spratte (@Skillmon) in Correct implementation of optional argument (comma-separated) in expl3.

\resetenumext The \resetenumext command *"resets"* the *start value* of the *"counters"* for the enumext and enumext*
__enumext_standard_reset:nn environments along with the *"internal counters"* used by the keys resume *without value* and resume* according
__enumext_starred_reset:n to the value of {⟨*some counter*⟩}.
__enumext_reset_count_resume:nn
__enumext_reset_count_resume:en
__enumext_reset_count_resume_all:n
__enumext_reset_count_resume_levels:n

```
2237  \NewDocumentCommand \resetenumext { s o m }
2238    {
2239        \bool_if:nTF {#1}
2240          {
2241              \__enumext_reset_count_resume_all:n {#3}
2242          }
2243          {
2244              \tl_if_novalue:nTF {#2}
2245                {
2246                    \__enumext_reset_count_resume_levels:n {#3}
2247                }
2248                {
```

```
2249              \str_if_eq:nnTF {#2} { * }
2250                  { \__enumext_starred_reset:n {#3} }
2251                  {
2252                    \bool_lazy_and:nnTF
2253                        { \int_compare_p:nNn {#2} > 0 }
2254                        { \int_compare_p:nNn {#2} < 5 }
2255                        { \__enumext_standard_reset:nn {#2} {#3} }
2256                        {
2257                          \msg_error:nne { enumext } { out-of-range } { \int_eval:n {#2} }
2258                        }
2259                  }
2260              }
2261          }
2262      }
2263  \cs_new_protected:Npn \__enumext_standard_reset:nn #1 %#2
2264      {
2265        \__enumext_reset_count_resume:en { \int_to_roman:n {#1} } %{#2}
2266      }
2267  \cs_new_protected:Npn \__enumext_starred_reset:n #1
2268      {
2269        \__enumext_reset_count_resume:nn { vii } {#1}
2270      }
2271  \cs_new_protected:Npn \__enumext_reset_count_resume:nn #1 #2
2272      {
2273        \counterwithin*{enumX#1}{#2}
2274        \counterwithin*{__enumext_resume_#1_int}{#2}
2275      }
2276  \cs_generate_variant:Nn \__enumext_reset_count_resume:nn { e }
2277  \cs_new_protected:Npn \__enumext_reset_count_resume_all:n #1
2278      {
2279        \clist_map_inline:nn { i,ii,iii,iv,vii }
2280            {
2281              \__enumext_reset_count_resume:nn { ##1 } { #1 }
2282            }
2283      }
2284  \cs_new_protected:Npn \__enumext_reset_count_resume_levels:n #1
2285      {
2286        \clist_map_inline:nn { i,ii,iii,iv }
2287            {
2288              \__enumext_reset_count_resume:nn { ##1 } { #1 }
2289            }
2290      }
```

(*End of definition for* \resetenumext *and others. This function is documented on page* *11.*)

13.28   The reset and reset* keys

The \resetenumext command does not work, for example, after an unnumbered chapter, so it is preferable to provide a pair of ⟨*keys*⟩ that adjust the internal variables if necessary.

reset
reset*
We define the keys reset and reset* for the *"all levels"* of enumext and enumext*.

```
2291  \cs_set_protected:Npn \__enumext_tmp:n #1
2292      {
2293        \keys_define:nn { enumext / #1 }
2294            {
2295              reset  .code:n = \__enumext_standard_reset_key:,
2296              reset  .value_forbidden:n = true,
2297              reset* .code:n = \__enumext_standard_reset_key_star:,
2298              reset* .value_forbidden:n = true,
2299            }
2300      }
2301  \clist_map_inline:nn {level-1, level-2, level-3, level-4} { \__enumext_tmp:n {#1} }
2302  \keys_define:nn { enumext / enumext* }
2303      {
2304        reset  .code:n = \__enumext_starred_reset_key:,
2305        reset  .value_forbidden:n = true,
2306        reset* .code:n = \__enumext_starred_reset_key:,
2307        reset* .value_forbidden:n = true,
2308      }
```

(*End of definition for* reset *and* reset*.)

13.28.1 Internal functions for `reset` and `reset*` keys

`\__enumext_standard_reset_key:`
`\__enumext_standard_reset_key_star:`
`\__enumext_starred_reset_key:`

The function `\__enumext_standard_reset_key:` will be handled by the `reset` key and will *"reset"* the counter `\c@__enumext_resume_X_int` to *"zero"* according to the *level* at which it is executed within the `enumext` environment.

```
2309  \cs_new_protected:Nn \__enumext_standard_reset_key:
2310    {
2311      \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2312        {
2313          \int_if_exist:cT { c@ __enumext_resume_ \int_to_roman:n { \l__enumext_level_int } _int }
2314            {
2315              \int_gzero:c { c@ __enumext_resume_ \int_to_roman:n { \l__enumext_level_int } _int }
2316            }
2317        }
2318    }
```

The function `\__enumext_standard_reset_key_star:` will be handled by the `reset*` key and will *"reset"* the counters `\c@__enumext_resume_X_int` to *"zero"* from the *level* at which it is executed within the `enumext` environment to the *lower levels*.

```
2319  \cs_new_protected:Nn \__enumext_standard_reset_key_star:
2320    {
2321      \cs_set:Npn \__enumext_tmp:n ##1
2322        {
2323          \int_if_exist:cT { c@ __enumext_resume_ \int_to_roman:n {##1} _int }
2324            {
2325              \int_gzero:c { c@ __enumext_resume_ \int_to_roman:n {##1} _int }
2326            }
2327        }
2328      \int_compare:nNnT { \l__enumext_level_int } > { 0 }
2329        {
2330          \int_step_function:nnN { \l__enumext_level_int } { 4 } \__enumext_tmp:n
2331        }
2332    }
```

The function `\__enumext_starred_reset_key:` will be handled by `reset` keys and `reset*` will *"reset"* the counter `\c@__enumext_resume_vii_int` to *"zero"* when executed in the `enumext*` environment.

```
2333  \cs_new_protected:Nn \__enumext_starred_reset_key:
2334    {
2335      \int_gzero:c { c@ __enumext_resume_vii_int }
2336    }
```

(*End of definition for* `\__enumext_standard_reset_key:` , `\__enumext_standard_reset_key_star:` , *and* `\__enumext_starred_-reset_key:`.)

13.29 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.

13.29.1 Setting `save-ans` key

`save-ans`

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

```
2337  \cs_set_protected:Npn \__enumext_tmp:n #1
2338    {
2339      \keys_define:nn { enumext / #1 }
2340        {
2341          save-ans .code:n = \__enumext_storing_set:n {##1},
2342          save-ans .value_required:n = true,
2343        }
2344    }
2345  \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

13.29.2 Internal functions for `save-ans` key

`\__enumext_start_save_ans_msg:`
`\__enumext_stop_save_ans_msg:`

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

```
2346  \cs_new_protected:Nn \__enumext_start_save_ans_msg:
2347    {
2348      \msg_term:nnVV { enumext } { save-ans-log }
```

```
2349        \g__enumext_envir_name_tl \l__enumext_store_name_tl
2350    }
2351 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
2352    {
2353      \msg_term:nnVV { enumext } { save-ans-log-hook }
2354        \g__enumext_envir_name_tl \g__enumext_store_name_tl
2355    }
```

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

__enumext_storing_set:n
__enumext_storing_exec:

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

```
2356 \cs_new_protected:Npn \__enumext_storing_set:n #1
2357    {
2358      \tl_set:Ne \l__enumext_store_name_tl {#1}
2359      \tl_if_empty:NTF \l__enumext_store_name_tl
2360        {
2361          \bool_lazy_or:nnT
2362            { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
2363            {
2364              \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
2365            }
2366        }
2367        {
2368          \bool_lazy_or:nnT
2369            { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
2370            {
2371              \__enumext_start_save_ans_msg:
2372              \__enumext_storing_exec:
2373            }
2374        }
2375    }
```

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.

```
2376 \cs_new_protected:Nn \__enumext_storing_exec:
2377    {
2378      \bool_set_true:N \l__enumext_store_active_bool
2379      \bool_set_true:N \l__enumext_check_answers_bool
2380      \tl_gset:NV \g__enumext_store_name_tl \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*.

```
2381        \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
2382          {
2383            \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
2384            \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
2385          }
2386        \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
2387          {
2388            \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
2389            \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
2390          }
2391        \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
2392          {
2393            \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
2394            \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
2395          }
2396    }
```

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

13.29.3 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.

13.29.4 Setting check-ans and no-store keys

check-ans
no-store

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

```
2397 \cs_set_protected:Npn \__enumext_tmp:n #1
2398   {
2399     \keys_define:nn { enumext / #1 }
2400       {
2401         check-ans .bool_set:N = \l__enumext_check_ans_key_bool,
2402         check-ans .initial:n  = false,
2403         check-ans .value_required:n = true,
2404         no-store  .code:n = {
2405                     \bool_set_false:N \l__enumext_check_answers_bool
2406                     \bool_set_false:N \l__enumext_check_ans_key_bool
2407                   },
2408         no-store  .value_forbidden:n = true,
2409       }
2410   }
2411 \clist_map_inline:nn
2412   {
2413     level-1, level-2, level-3, level-4, enumext*
2414   }
2415   { \__enumext_tmp:n {#1} }
```

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

13.29.5 Set-up check answer mechanism

__enumext_check_ans_active:
__enumext_check_ans_level:

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.

```
2416 \cs_new_protected:Nn \__enumext_check_ans_active:
2417   {
2418     \tl_if_empty:NF \l__enumext_store_name_tl
2419       {
2420         \bool_if:NT \l__enumext_check_answers_bool
2421           {
2422             \__enumext_check_ans_level:
2423           }
2424       }
2425   }
```

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"* .

```
2426  \cs_new_protected:Nn \__enumext_check_ans_level:
2427    {
2428      \int_case:nn { \l__enumext_level_int }
2429        {
2430          { 1 }{
2431              \bool_lazy_all:nT
2432                {
2433                  { \bool_if_p:N \g__enumext_starred_bool }
2434                  { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
2435                }
2436                {
2437                  \int_gdecr:N \g__enumext_item_number_int
2438                  \bool_set_false:N \l__enumext_item_number_bool
2439                }
2440            }
2441          { 2 }{
2442              \int_gdecr:N \g__enumext_item_number_int
2443              \bool_set_false:N \l__enumext_item_number_bool
2444            }
2445          { 3 }{
2446              \int_gdecr:N \g__enumext_item_number_int
2447              \bool_set_false:N \l__enumext_item_number_bool
2448            }
2449          { 4 }{
2450              \int_gdecr:N \g__enumext_item_number_int
2451              \bool_set_false:N \l__enumext_item_number_bool
2452            }
2453        }
```

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.

```
2454      \int_case:nn { \l__enumext_level_h_int }
2455        {
2456          { 1 }{
2457              \bool_lazy_all:nT
2458                {
2459                  { \bool_if_p:N \g__enumext_standar_bool }
2460                  { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
2461                }
2462                {
2463                  \int_gdecr:N \g__enumext_item_number_int
2464                  \bool_set_false:N \l__enumext_item_number_bool
2465                }
2466            }
2467        }
2468    }
```

(*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.

```
2469  \cs_new_protected:Nn \__enumext_check_ans_key_hook:
2470    {
2471      \bool_lazy_and:nnT
2472        { \bool_if_p:N \l__enumext_check_ans_key_bool }
2473        { \bool_if_p:N \g__enumext_standar_bool }
2474        {
2475          \bool_gset_true:N \g__enumext_check_ans_key_bool
2476        }
2477      \bool_lazy_and:nnT
2478        { \bool_if_p:N \l__enumext_check_ans_key_bool }
2479        { \bool_if_p:N \g__enumext_starred_bool }
2480        {
2481          \bool_gset_true:N \g__enumext_check_ans_key_bool
2482        }
2483    }
```

(*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:.

```
2484 \cs_new_protected:Nn \__enumext_item_answer_diff:
2485   {
2486     \int_gset:Nn \g__enumext_item_answer_diff_int
2487       {
2488         \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
2489       }
2490   }
```

(*End of definition for* __enumext_item_answer_diff:*.*)

__enumext_check_ans_show:
__enumext_check_ans_msg_less:
__enumext_check_ans_msg_same_ok:
__enumext_check_ans_msg_greater:

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

```
2491 \cs_new_protected:Nn \__enumext_check_ans_show:
2492   {
2493     \int_case:nn { \g__enumext_item_answer_diff_int }
2494       {
2495         { -1 }{ \__enumext_check_ans_msg_less:      }
2496         {  0 }{ \__enumext_check_ans_msg_same_ok: }
2497         {  1 }{ \__enumext_check_ans_msg_greater: }
2498       }
2499   }
2500 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
2501   {
2502     \msg_warning:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2503       { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2504   }
2505 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
2506   {
2507     \msg_term:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2508       { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2509   }
2510 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
2511   {
2512     \msg_warning:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2513       { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2514   }
```

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

__enumext_check_ans_log:
__enumext_check_ans_log_msg_less:
__enumext_check_ans_log_msg_same_ok:
__enumext_check_ans_log_msg_greater:

The function __enumext_check_ans_log: will be executed within the function __enumext_execute_-after_env: when the key check-ans is not active, that is, when \g__enumext_check_ans_key_bool is *"false"* and write in the log the appropriate message according to the value of \g__enumext_item_answer_-diff_int set by the function __enumext_item_answer_diff:.

```
2515 \cs_new_protected:Nn \__enumext_check_ans_log:
2516   {
2517     \int_case:nn { \g__enumext_item_answer_diff_int }
2518       {
2519         { -1 }{ \__enumext_check_ans_log_msg_less:      }
2520         {  0 }{ \__enumext_check_ans_log_msg_same_ok: }
2521         {  1 }{ \__enumext_check_ans_log_msg_greater: }
2522       }
2523   }
2524 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
2525   {
2526     \msg_log:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2527       { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2528   }
2529 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
2530   {
2531     \msg_log:nneee { enumext } { items-same-answer }  { \g__enumext_store_name_tl }
2532       { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2533   }
```

```
2534 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
2535   {
2536     \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2537       { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2538   }
```

(*End of definition for* __enumext_check_ans_log: *and others.*)

13.29.6 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.

```
2539 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
2540   {
2541     \int_compare:nNnT
2542       { \g__enumext_check_starred_cmd_int } = { 0 }
2543       {
2544         \msg_warning:nnnV
2545           { enumext } { missing-starred }{ #1 } \l__enumext_check_start_line_env_tl
2546       }
2547     \int_compare:nNnT
2548       { \g__enumext_check_starred_cmd_int } > { 1 }
2549       {
2550         \msg_warning:nnnV
2551           { enumext } { many-starred }{ #1 } \l__enumext_check_start_line_env_tl
2552       }
2553     \int_gzero:N \g__enumext_check_starred_cmd_int
2554     \tl_clear:N \l__enumext_check_start_line_env_tl
2555   }
```

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

13.30 Keys and functions associated with storage

13.30.1 Keys for marks, wrap and show

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

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

```
2556 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2557   {
2558     \keys_define:nn { enumext / #1 }
2559       {
2560         mark-ans*  .tl_set:c  = { l__enumext_mark_answer_sym_#2_tl },
2561         mark-ans*  .initial:n = \textasteriskcentered,
2562         mark-ans*  .value_required:n = true,
2563         mark-pos*  .choice:,
2564         mark-pos* / left    .code:n = \str_set:cn { l__enumext_mark_position_#2_str } { l },
2565         mark-pos* / right   .code:n = \str_set:cn { l__enumext_mark_position_#2_str } { r },
2566         mark-pos* / center  .code:n = \str_set:cn { l__enumext_mark_position_#2_str } { c },
2567         mark-pos* / unknown .code:n =
2568                             \msg_error:nneee { enumext } { unknown-choice }
2569                               { mark-pos } { left,~right,~center } { \exp_not:n {##1} },
2570         mark-pos*  .initial:n = right,
2571         mark-pos*  .value_required:n = true,
2572         mark-sep*  .dim_set:c = { l__enumext_mark_sym_sep_#2_dim },
2573         mark-sep*  .value_required:n = true,
2574         wrap-ans*  .cs_set_protected:cp = { __enumext_keyans_wrapper_item_#2:n } ##1,
2575         wrap-ans*  .value_required:n = true,
2576         wrap-opt   .cs_set_protected:cp = { __enumext_keyans_wrapper_opt_#2:n } ##1,
2577         wrap-opt   .initial:n = [{##1}],
2578         wrap-opt   .value_required:n = true,
2579         save-sep   .tl_set:c = { l__enumext_store_keyans_item_opt_sep_#2_tl },
2580         save-sep   .initial:n = {,~},
2581         save-sep   .value_required:n = true,
2582         show-ans   .bool_set:N = \l__enumext_show_answer_bool,
2583         show-ans   .initial:n = false,
2584         show-ans   .value_required:n = true,
2585         show-pos   .bool_set:N = \l__enumext_show_position_bool,
2586         show-pos   .initial:n = false,
```

```
2587              show-pos    .value_required:n = true,
2588          }
2589      }
2590  \clist_map_inline:nn { {keyans}{v}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }
```

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

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

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

```
2591  \cs_set_protected:Npn \__enumext_tmp:n #1
2592    {
2593      \keys_define:nn { enumext / #1 }
2594        {
2595          mark-ref  .tl_set:N   = \l__enumext_mark_ref_sym_tl,
2596          mark-ref  .initial:n  = \textreferencemark,
2597          mark-ref  .value_required:n = true,
2598          save-ref  .bool_set:N = \l__enumext_store_ref_key_bool,
2599          save-ref  .initial:n  = false,
2600          save-ref  .value_required:n = true,
2601          show-ans  .bool_set:N = \l__enumext_show_answer_bool,
2602          show-ans  .initial:n  = false,
2603          show-ans  .value_required:n = true,
2604          show-pos  .bool_set:N = \l__enumext_show_position_bool,
2605          show-pos  .initial:n  = false,
2606          show-pos  .value_required:n = true,
2607          mark-ans  .tl_set:N   = \l__enumext_mark_answer_sym_tl,
2608          mark-ans  .initial:n = \textasteriskcentered,
2609          mark-ans  .value_required:n = true,
2610          mark-sep  .dim_set:N  = \l__enumext_mark_sym_sep_dim,
2611          mark-sep  .value_required:n = true,
2612          mark-pos  .choice:,
2613          mark-pos / left    .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2614          mark-pos / right   .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2615          mark-pos / center  .code:n = \str_set:Nn \l__enumext_mark_position_str { c },
2616          mark-pos / unknown .code:n =
2617                          \msg_error:nneee { enumext } { unknown-choice }
2618                            { mark-pos } { left,~right,~center } { \exp_not:n {##1} },
2619          mark-pos  .initial:n = right,
2620          mark-pos  .value_required:n = true,
2621
2622          wrap-ans  .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
2623          wrap-ans  .initial:n =
2624                      {
2625                        \fbox{\parbox[t]{\dimeval{\itemwidth -2\fboxsep -2\fboxrule}}{##1}}
2626                      },
2627          wrap-ans  .value_required:n = true,
2628          mark-ans* .code:n = {
2629                              \keys_set:nn { enumext / keyans  } { mark-ans* = {##1} }
2630                              \keys_set:nn { enumext / keyans* } { mark-ans* = {##1} }
2631                            },
2632          mark-ans* .value_required:n = true,
2633          mark-pos* .code:n = {
2634                              \keys_set:nn { enumext / keyans  } { mark-pos* = {##1} }
2635                              \keys_set:nn { enumext / keyans* } { mark-pos* = {##1} }
2636                            },
2637          mark-pos* .value_required:n = true,
2638          mark-sep* .code:n = {
2639                              \keys_set:nn { enumext / keyans  } { mark-sep* = {##1} }
2640                              \keys_set:nn { enumext / keyans* } { mark-sep* = {##1} }
2641                            },
2642          mark-sep* .value_required:n = true,
2643          wrap-ans* .code:n = {
2644                              \keys_set:nn { enumext / keyans  } { wrap-ans* = {##1} }
2645                              \keys_set:nn { enumext / keyans* } { wrap-ans* = {##1} }
2646                            },
2647          wrap-ans* .value_required:n = true,
2648          wrap-opt  .code:n = {
2649                              \keys_set:nn { enumext / keyans  } { wrap-opt = {##1} }
2650                              \keys_set:nn { enumext / keyans* } { wrap-opt = {##1} }
```

```
2651                                   },
2652          wrap-opt  .value_required:n = true,
2653          save-sep  .code:n = {
2654                                   \keys_set:nn { enumext / keyans  } { save-sep = {##1} }
2655                                   \keys_set:nn { enumext / keyans* } { save-sep = {##1} }
2656                                   },
2657          save-sep  .value_required:n = true,
2658        }
2659    }
2660 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

13.30.2 Storing structure of the environments

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

`\__enumext_store_active_keys:n`
`\__enumext_store_active_keys_vii:n`

The functions `\__enumext_store_active_keys:n` and `\__enumext_store_active_keys_vii:n` will be responsible for the *"storing keys"* filtered from the *optional argument* of the environment in which the key `save-ans` is executed and the levels within this for the `enumext` and `enumext*` environments. We will execute this function only if the variable `\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*⟩.

```
2661 \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2662   {
2663     \bool_if:cF { l__enumext_store_save_key_ \__enumext_level: _bool }
2664       {
2665         \tl_clear:c { l__enumext_store_save_key_ \__enumext_level: _tl }
2666         \tl_set:ce
2667           { l__enumext_store_save_key_ \__enumext_level: _tl }
2668           { \__enumext_filter_save_key:n {#1} }
2669       }
2670   }
2671 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2672   {
2673     \bool_if:NF \l__enumext_store_save_key_vii_bool
2674       {
2675         \tl_clear:N \l__enumext_store_save_key_vii_tl
2676         \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2677       }
2678   }
```

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

13.30.3 Setting save-key key

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

`save-key`

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

```
2679 \cs_set_protected:Npn \__enumext_tmp:n #1
2680   {
2681     \keys_define:nn { enumext / enumext* }
2682       {
2683         save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2684         save-key .value_required:n = true,
2685       }
2686     \keys_define:nn { enumext / #1 }
2687       {
2688         save-key .code:n = \__enumext_parse_save_key:n {##1},
2689         save-key .value_required:n = true,
2690       }
2691   }
2692 \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*.

```
2693 \cs_new_protected:Npn \__enumext_parse_save_key:n #1
2694   {
2695     \bool_set_true:c { l__enumext_store_save_key_ \__enumext_level: _bool }
2696     \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2697     \tl_set:ce
2698       { l__enumext_store_save_key_ \__enumext_level: _tl }
2699       { \__enumext_filter_save_key:n {#1} }
2700   }
2701 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2702   {
2703     \bool_set_true:N \l__enumext_store_save_key_vii_bool
2704     \tl_clear:N \l__enumext_store_save_key_vii_tl
2705     \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2706   }
```

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

13.30.4 Internal functions to store optional arguments

__enumext_filter_save_key:n
__enumext_filter_save_key_key:n
__enumext_filter_save_key_pair:nn

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

```
2707 \cs_new:Npn \__enumext_filter_save_key:n #1
2708   {
2709     \use:e
2710       {
2711         \keyval_parse:NNn
2712           \__enumext_filter_save_key_key:n
2713           \__enumext_filter_save_key_pair:nn {#1}
2714       }
2715   }
```

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

```
2716 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2717   {
2718     \str_case:nnF {#1}
2719       {
2720         { resume } {} { resume* } {} { reset } {} { reset* } {} { no-store } {} { base-fix } {}
2721       }
2722       { , { \exp_not:n {#1} } }
2723   }
```

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

```
2724 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2725   {
2726     \str_case:nnF {#1}
2727       {
2728         { series   } {} { resume     } {} { save-ans   } {} { save-ref  } {}
2729         { save-key  } {} { check-ans  } {} { show-ans   } {} { show-pos  } {}
2730         { mark-ans  } {} { mark-pos   } {} { mark-sep   } {} { wrap-ans  } {}
2731         { mark-ans* } {} { mark-pos*  } {} { mark-sep*  } {} { wrap-ans* } {}
2732         { wrap-opt  } {} { save-sep   } {} { mark-ref   } {} { mini-env  } {}
2733         { mini-sep  } {} { mini-right } {} { mini-right* } {}
2734       }
2735       { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2736   }
```

(*End of definition for* __enumext_filter_save_key:n, __enumext_filter_save_key_key:n, *and* __enumext_filter_-save_key_pair:nn.)

13.30.5 Function for storing content in prop list

__enumext_store_addto_prop:n
__enumext_store_addto_prop:V

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

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

```
2737 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
```

```
2738    {
2739      \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2740        {
2741          \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2742        }
2743        { #1 }
2744    }
2745  \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V }
```

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

13.30.6   Function for storing content in sequence

__enumext_store_addto_seq:n
__enumext_store_addto_seq:v
__enumext_store_addto_seq:V

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

The form in which the {⟨*content*⟩} is stored in *sequence* is in 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.

```
2746  \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2747    {
2748      \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
2749    }
2750  \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V }
```

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

13.30.7   Functions for storing structure in the sequence

__enumext_store_level_open:
__enumext_store_level_close:

The *"storing structure"* is handled by the functions __enumext_store_level_open: and __enumext_-store_level_close: which are executed per level within the enumext environment.

```
2751  \cs_new_protected:Nn \__enumext_store_level_open:
2752    {
2753      \bool_if:NT \l__enumext_check_answers_bool
2754        {
2755          \tl_if_empty:cTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2756            {
2757              \__enumext_store_addto_seq:n
2758                {
2759                  \item \begin{enumext}
2760                }
2761            }
2762            {
2763              \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2764                {
2765                  \item \begin{enumext} [
2766                }
2767              \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2768                {
2769                  ]
2770                }
2771              \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2772            }
2773        }
2774    }
2775  \cs_new_protected:Nn \__enumext_store_level_close:
2776    {
2777      \bool_if:NT \l__enumext_check_answers_bool
2778        {
2779          \__enumext_store_addto_seq:n { \end{enumext} }
2780        }
2781    }
```

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

__enumext_store_level_open_vii:
__enumext_store_level_close_vii:

The *"storing structure"* is handled by the functions __enumext_store_level_open_vii: and __enumext_-store_level_close_vii: which are executed in the enumext* environment.

```
2782  \cs_new_protected:Nn \__enumext_store_level_open_vii:
2783    {
2784      \bool_if:NT \l__enumext_check_answers_bool
2785        {
2786          \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2787            {
```

```
2788            \__enumext_store_addto_seq:n
2789              {
2790                \item \begin{enumext*}
2791              }
2792          }
2793          {
2794            \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2795              {
2796                \item \begin{enumext*}[
2797              }
2798            \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2799              {
2800                ]
2801              }
2802            \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2803          }
2804      }
2805    }
2806  \cs_new_protected:Nn \__enumext_store_level_close_vii:
2807    {
2808      \bool_if:NT \l__enumext_check_answers_bool
2809        {
2810          \__enumext_store_addto_seq:n { \end{enumext*} }
2811        }
2812    }
```

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

13.30.8 Function for show marks and position

__enumext_print_keyans_box:NN
__enumext_print_keyans_box:cc

The function __enumext_print_keyans_box:NN print a box in the left margin with \l__enumext_mark_-answer_sym_tl used by the wrap-ans, show-ans and show-pos keys. The function takes two arguments:

#1 : \l__enumext_labelwidth_X_dim
#2 : \l__enumext_labelsep_X_dim

```
2813  \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2814    {
2815      \mode_leave_vertical:
2816      \skip_horizontal:n { -\dim_use:N #2 }
2817      \hbox_overlap_left:n
2818        {
2819          \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2820            {
2821              \tl_use:N \l__enumext_mark_answer_sym_tl
2822            }
2823        }
2824      \skip_horizontal:n { \dim_use:N #2 }
2825    }
2826  \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }
```

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

13.31 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.

```
2827  \cs_new_protected:Nn \__enumext_store_internal_ref:
2828    {
2829      \cs_set_protected:Npn \__enumext_tmp:n ##1
2830        {
2831          \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2832          \tl_reverse:c { l__enumext_label_copy_##1_tl }
2833          \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2834          \tl_reverse:c { l__enumext_label_copy_##1_tl }
2835        }
2836      \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2837      \cs_set:Npn \__enumext_tmp:n ##1
2838        { . \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.

```
2839        \bool_lazy_all:nT
2840          {
2841            { \bool_if_p:N \g__enumext_starred_bool }
2842            { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2843          }
2844          {
2845            \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2846              { \tl_use:N \l__enumext_label_copy_vii_tl }
2847          }
2848        \bool_lazy_all:nT
2849          {
2850            { \bool_not_p:n { \g__enumext_standar_bool } }
2851            { \bool_if_p:N \l__enumext_standar_bool }
2852            { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2853          }
2854          {
2855            \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2856              {
2857                \tl_use:N \l__enumext_label_copy_vii_tl
2858                \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2859              }
2860          }
```

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.

```
2861        \bool_lazy_all:nT
2862          {
2863            { \bool_if_p:N \g__enumext_standar_bool }
2864            { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2865            { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2866          }
2867          {
2868            \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2869              {
2870                \tl_use:N \l__enumext_label_copy_i_tl
2871                \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2872              }
2873          }
2874        \cs_set:Npn \__enumext_tmp:n ##1
2875          { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } . }
2876        \bool_lazy_all:nT
2877          {
2878            { \bool_if_p:N \g__enumext_standar_bool }
2879            { \bool_if_p:N \l__enumext_starred_bool }
2880            { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2881          }
2882          {
2883            \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2884              {
2885                \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2886                \tl_use:N \l__enumext_label_copy_vii_tl
2887              }
2888          }
```

Now we set the variable \l__enumext_newlabel_arg_one_tl which will contain {⟨*store name* : *position*⟩}.

```
2889        \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2890          {
2891            \l__enumext_store_name_tl \c_colon_str
2892            \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2893          }
```

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.

```
2894        \tl_put_right:Ne \l__enumext_write_aux_file_tl
2895          {
2896            \__enumext_newlabel:nn
2897              { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2898              { \l__enumext_newlabel_arg_two_tl }
2899          }
2900        \l__enumext_write_aux_file_tl
```

2901   }

(*End of definition for* __enumext_store_internal_ref:.)

13.32 Common functions for \anskey and anskey* environment

__enumext_store_anskey_arg:n

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

```
2902  \cs_new_protected:Npn \__enumext_store_anskey_arg:n #1
2903    {
2904      \int_gincr:N \g__enumext_item_anskey_int
2905      \__enumext_store_addto_prop:n {#1}
2906      \bool_if:NT \l__enumext_store_ref_key_bool
2907        {
2908          \__enumext_store_internal_ref:
2909        }
2910      \__enumext_anskey_show_wrap_left:n { #1 }
```

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

```
2911      \tl_clear:N \l__enumext_store_anskey_arg_tl
2912      \bool_lazy_and:nnT
2913        { \bool_if_p:N \l__enumext_store_columns_break_bool }
2914        { \bool_not_p:n { \l__enumext_starred_bool } }
2915        {
2916          \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2917        }
2918      \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { \item }
```

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

```
2919      \bool_lazy_and:nnT
2920        { \bool_not_p:n { \l__enumext_starred_bool } }
2921        { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2922        {
2923          \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2924            {
2925              ( \exp_not:V \l__enumext_store_item_join_int )
2926            }
2927        }
```

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*.

```
2928      \bool_if:NTF \l__enumext_store_item_star_bool
2929        {
2930          \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2931          \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2932            {
2933              \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2934                {
2935                  [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2936                }
2937            }
2938          \dim_compare:nT
2939            {
2940              \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2941            }
2942            {
2943              \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2944                {
2945                  [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2946                }
2947            }
2948          \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2949        }
2950        {
2951          \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2952        }
```

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*.

```
2953    \bool_lazy_and:nnT
2954      { \bool_if_p:N \l__enumext_store_ref_key_bool }
2955      { \bool_if_p:N \l__enumext_hyperref_bool }
2956      {
2957        \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2958          {
2959            \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2960                { \exp_not:V \l__enumext_mark_ref_sym_tl }
2961          }
2962      }
2963    \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2964  }
```

(*End of definition for* `\__enumext_store_anskey_arg:n`.)

`\__enumext_anskey_show_wrap_arg:n` The function `\__enumext_anskey_show_wrap_arg:n` *"wraps"* the {⟨*argument*⟩} passed to `\anskey` and the ⟨*body*⟩ for `anskey*` when using the `wrap-ans` and `wrap-sep` keys.

```
2965  \cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
2966    {
2967      \par
2968      \bool_if:NTF \l__enumext_starred_bool
2969        {
2970          \dim_compare:nNnT { \l__enumext_mark_sym_sep_dim } = { \c_zero_dim }
2971            {
2972              \dim_set:Nn \l__enumext_mark_sym_sep_dim { \l__enumext_labelsep_vii_dim }
2973            }
2974          \__enumext_print_keyans_box:NN
2975            \l__enumext_labelwidth_vii_dim \l__enumext_mark_sym_sep_dim
2976        }
2977        {
2978          \dim_compare:nNnT { \l__enumext_mark_sym_sep_dim } = { \c_zero_dim }
2979            {
2980              \dim_set:Nn \l__enumext_mark_sym_sep_dim
2981                {
2982                  \dim_use:c {l__enumext_labelsep_ \__enumext_level: _dim }
2983                }
2984            }
2985          \__enumext_print_keyans_box:cc
2986            { l__enumext_labelwidth_ \__enumext_level: _dim } { l__enumext_mark_sym_sep_dim }
2987        }
2988      \__enumext_anskey_wrapper:n { #1 }
2989    }
```

(*End of definition for* `\__enumext_anskey_show_wrap_arg:n`.)

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

```
2990  \cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
2991    {
2992      \bool_if:NT \l__enumext_show_answer_bool
2993        {
2994          \__enumext_anskey_show_wrap_arg:n { #1 }
2995        }
2996      \bool_if:NT \l__enumext_show_position_bool
2997        {
2998          \tl_set:Ne \l__enumext_mark_answer_sym_tl
2999            {
3000              \group_begin:
3001              \exp_not:N \normalfont
3002              \exp_not:N \footnotesize [ \int_eval:n
3003                {
3004                  \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3005                }
3006              ]
3007              \group_end:
3008            }
```

```
3009            \__enumext_anskey_show_wrap_arg:n { #1 }
3010        }
3011    }
```

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

13.33 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*⟩}.

break-col
item-join
item-star
item-sym*
item-pos*
unknown
__enumext_anskey_unknown:n
__enumext_anskey_unknown:nn

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

```
3012 \keys_define:nn { enumext / anskey }
3013    {
3014      break-col .bool_set:N = \l__enumext_store_columns_break_bool,
3015      break-col .default:n  = true,
3016      break-col .value_forbidden:n = true,
3017      item-join .int_set:N  = \l__enumext_store_item_join_int,
3018      item-join .value_required:n = true,
3019      item-star .bool_set:N = \l__enumext_store_item_star_bool,
3020      item-star .default:n  = true,
3021      item-star .value_forbidden:n = true,
3022      item-sym* .tl_set:N   = \l__enumext_store_item_symbol_tl,
3023      item-sym* .value_required:n = true,
3024      item-pos* .dim_set:N  = \l__enumext_store_item_symbol_sep_dim,
3025      item-pos* .value_required:n = true,
3026      unknown   .code:n     = { \__enumext_anskey_unknown:n {#1} },
3027    }
```

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.

```
3028 \cs_new_protected:Npn \__enumext_anskey_unknown:n #1
3029    {
3030      \exp_args:NV \__enumext_anskey_unknown:nn \l_keys_key_str {#1}
3031    }
3032 \cs_new_protected:Npn \__enumext_anskey_unknown:nn #1 #2
3033    {
3034      \tl_if_blank:nTF {#2}
3035        {
3036          \msg_error:nnn { enumext } { anskey-cmd-key-unknown } {#1}
3037        }
3038        {
3039          \msg_error:nnnn { enumext } { anskey-cmd-key-value-unknown } {#1} {#2}
3040        }
3041    }
```

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

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

\anskey

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

```
3042 \NewDocumentCommand \anskey { o +m }
3043    {
3044      \__enumext_anskey_safe_outer:
3045      \group_begin:
3046        \bool_if:NT \l__enumext_check_answers_bool
3047          {
3048            \tl_if_novalue:nF {#1}
3049              {
3050                \keys_set:nn { enumext / anskey } {#1}
3051              }
3052            \tl_if_blank:nTF {#2}
3053              {
3054                \msg_error:nn { enumext } { anskey-empty-arg }
```

```
3055                }
3056              {
3057                \__enumext_anskey_safe_inner:
3058                \__enumext_store_anskey_arg:n {#2}
3059              }
3060            }
3061        \group_end:
3062      }
```

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

13.33.1 Internal functions for the command

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

```
3063  \cs_new_protected:Nn \__enumext_anskey_safe_outer:
3064    {
3065      \bool_if:NF \l__enumext_store_active_bool
3066        {
3067          \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
3068        }
3069      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
3070        {
3071          \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
3072        }
3073      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
3074        {
3075          \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans* }
3076        }
3077      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
3078        {
3079          \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
3080        }
3081    }
```

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.

```
3082  \cs_new_protected:Nn \__enumext_anskey_safe_inner:
3083    {
3084      \int_incr:N \l__enumext_anskey_level_int
3085      \int_compare:nNnT { \l__enumext_anskey_level_int } > { 1 }
3086        {
3087          \msg_error:nn { enumext } { anskey-nested }
3088        }
3089      \bool_if:NF \l__enumext_item_number_bool
3090        {
3091          \msg_error:nn { enumext } { anskey-unnumber-item }
3092        }
3093      \mode_if_math:T
3094        {
3095          \msg_error:nne { enumext } { anskey-math-mode } { \c_backslash_str anskey }
3096        }
3097    }
```

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

13.34 The environment anskey*

The original implementation of the anskey* environment used non-public functions from the scontents[4]
package, which was not the best approach. Fortunately LaTeX release 2025-06-01 implemented the new c-type
argument in the ltcmd[13], with which we can record the ⟨body⟩ of the environment in *verbatim mode* and
\tl_retokenize:n (wrapper around the \scantokens) provide by LaTeX3 release 2025-07-08 do the work as
the original implementation.

break-col First we add the same keys from the \anskey command along with the force-eol, write-env and
item-join overwrite keys that were in the original implementation that used the scontents support package for
item-star these.
item-sym*
item-pos*
force-eol
write-env
overwrite
unknown

```
3098  \keys_define:nn { enumext / anskey* }
3099    {
3100      break-col .bool_set:N = \l__enumext_store_columns_break_bool,
3101      break-col .default:n  = true,
3102      break-col .value_forbidden:n = true,
3103      item-join .int_set:N  = \l__enumext_store_item_join_int,
```

```
3104      item-join .value_required:n = true,
3105      item-star .bool_set:N = \l__enumext_store_item_star_bool,
3106      item-star .default:n  = true,
3107      item-star .value_forbidden:n = true,
3108      item-sym* .tl_set:N    = \l__enumext_store_item_symbol_tl,
3109      item-sym* .value_required:n = true,
3110      item-pos* .dim_set:N  = \l__enumext_store_item_symbol_sep_dim,
3111      item-pos* .value_required:n = true,
3112      force-eol .bool_set:N = \l__enumext_anskey_env_force_eol_bool,
3113      force-eol .initial:n  = false,
3114      force-eol .default:n  = true,
3115      write-env .code:n     = {
3116                             \bool_set_true:N \l__enumext_write_anskey_env_bool
3117                             \tl_set:Nn \l__enumext_write_anskey_env_file_name_tl {#1}
3118                          },
3119      write-env .value_required:n = true,
3120      overwrite .bool_set:N = \l__enumext_anskey_env_overwrite_bool,
3121      overwrite .initial:n  = false,
3122      overwrite .default:n  = true,
3123      unknown   .code:n     = { \__enumext_anskey_env_unknown:n {#1} },
3124    }
```

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

__enumext_anskey_env_unknown:n
__enumext_anskey_env_unknown:nn

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

```
3125 \cs_new_protected:Npn \__enumext_anskey_env_unknown:n #1
3126   {
3127     \exp_args:NV \__enumext_anskey_env_unknown:nn \l_keys_key_str {#1}
3128   }
3129 \cs_new_protected:Npn \__enumext_anskey_env_unknown:nn #1#2
3130   {
3131     \tl_if_blank:nTF {#2}
3132       {
3133         \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
3134       }
3135       {
3136         \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
3137       }
3138   }
```

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

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

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

```
3139 \prg_new_protected_conditional:Npnn \__enumext_anskey_env_file_if_writable:n #1 { T, F, TF }
3140   {
3141     \bool_if:NTF \l__enumext_write_anskey_env_bool
3142       {
3143         \file_if_exist:nTF {#1}
3144           {
3145             \bool_if:NTF \l__enumext_anskey_env_overwrite_bool
3146               {
3147                 \msg_warning:nne { enumext } { overwrite-file } {#1}
3148                 \prg_return_true:
3149               }
3150               {
3151                 \msg_warning:nne { enumext } { not-writing } {#1}
3152                 \prg_return_false:
3153               }
3154           }
3155           {
3156             \msg_warning:nne { enumext } { writing-file } {#1}
3157             \prg_return_true:
3158           }
3159       }
3160       { \prg_return_false: }
3161   }
```

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

```
3162  \cs_new_protected:Npn \__enumext_anskey_env_file_write:nn #1#2
3163    {
3164      \__enumext_anskey_env_file_if_writable:nT {#1}
3165        {
3166          \iow_open:Nn \l__enumext_write_anskey_env_file_iow {#1}
3167          \iow_now:Nn  \l__enumext_write_anskey_env_file_iow {#2}
3168          \iow_close:N \l__enumext_write_anskey_env_file_iow
3169        }
3170    }
3171  \cs_generate_variant:Nn \__enumext_anskey_env_file_write:nn { VV }
```

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

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

```
3172  \NewDocumentEnvironment{anskey*}{ o c }
3173    {
3174      \__enumext_anskey_env_safe_outer:
3175      \bool_if:NT \l__enumext_check_answers_bool
3176        {
3177          \tl_if_novalue:nF {#1}
3178            {
3179              \keys_set:nn { enumext / anskey* } {#1}
3180            }
3181          \tl_if_blank:nTF {#2}
3182            {
3183              \msg_error:nn { enumext } { anskey-empty-arg }
3184            }
3185            {
3186              \__enumext_anskey_env_safe_inner:
3187              \__enumext_store_anskey_env:n {#2}
3188            }
3189        }
3190    } { }
```

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

13.34.1   Internal functions for the environment

__enumext_anskey_env_safe_outer:
__enumext_anskey_env_safe_inner:
__enumext_store_anskey_env:n

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

```
3191  \cs_new_protected:Nn \__enumext_anskey_env_safe_outer:
3192    {
3193      \bool_if:NF \l__enumext_store_active_bool
3194        {
3195          \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
3196        }
3197      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
3198        {
3199          \msg_error:nnn { enumext } { anskey-env-wrong }{ keyans }
3200        }
3201      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
3202        {
3203          \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
3204        }
3205      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
3206        {
3207          \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
3208        }
3209    }
```

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

```
3210  \cs_new_protected:Nn \__enumext_anskey_env_safe_inner:
3211    {
3212      \bool_if:NF \l__enumext_item_number_bool
3213        {
3214          \msg_error:nn { enumext } { anskey-unnumber-item }
```

```
3215          }
3216       \mode_if_math:T
3217          {
3218             \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
3219          }
3220       }
```

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

```
3221  \cs_new_protected:Npn \__enumext_store_anskey_env:n #1
3222     {
3223       \tl_set:Nn \l__enumext_store_anskey_env_tl {#1}
3224       \RenewDocumentCommand \obeyedline { } { \iow_char:N \^^J }
3225       \tl_replace_all:Nee \l__enumext_store_anskey_env_tl { \obeyedline } { \iow_char:N \^^J }
3226       \__enumext_anskey_env_file_write:VV
3227          \l__enumext_write_anskey_env_file_name_tl \l__enumext_store_anskey_env_tl
3228       \bool_if:NF \l__enumext_anskey_env_force_eol_bool
3229          {
3230             \tl_put_right:Ne \l__enumext_store_anskey_env_tl
3231                {
3232                   \c__enumext_anskey_env_hidden_space_str
3233                }
3234          }
3235       \exp_args:Ne
3236          \__enumext_store_anskey_arg:n
3237             {
3238                \tl_retokenize:n { \l__enumext_store_anskey_env_tl }
3239             }
3240     }
```

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

(*End of definition for* __enumext_anskey_env_safe_outer: , __enumext_anskey_env_safe_inner: , *and* __enumext_store_-anskey_env:n.)

13.35   Executing check-ans system and write .log

__enumext_execute_after_env:

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

```
3241  \cs_new_protected:Nn \__enumext_execute_after_env:
3242     {
3243       \int_compare:nNnT { \l__enumext_level_int } = { 0 }
3244          {
3245             \tl_if_empty:NF \g__enumext_store_name_tl
3246                {
3247                   \__enumext_stop_save_ans_msg:
3248                   \__enumext_item_answer_diff:
3249                   \__enumext_log_global_vars:
3250                   \__enumext_log_answer_vars:
3251                   \bool_if:NTF \g__enumext_check_ans_key_bool
3252                      {
3253                         \__enumext_check_ans_show:
3254                      }
3255                      { \__enumext_check_ans_log: }
3256                }
3257             \__enumext_reset_global_vars:
3258          }
3259     }
```

🖋 This function is passed to the function __enumext_after_env:nn for the environments enumext (§13.42) and enumext* (§13.47) and it is executed only when the environments are not nested or at some level of these..

(*End of definition for* __enumext_execute_after_env:*.*)

13.36 Common functions for keyans, keyans* and keyanspic

13.36.1 Storing content in prop list

__enumext_keyans_addto_prop:n

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

```
3260  \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
3261    {
3262      \tl_clear:N \l__enumext_store_current_label_tl
3263      \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
3264        {
3265          \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
3266        }
3267        {
3268          \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
3269        }
```

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

```
3270      \tl_if_novalue:nF { #1 }
3271        {
3272          \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_v_tl
3273            {
3274              \tl_put_right:NV \l__enumext_store_current_label_tl \l__enumext_store_keyans_item_opt
3275            }
3276          \tl_put_right:Nn \l__enumext_store_current_label_tl { #1 }
3277        }
3278      \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
3279    }
```

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

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

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

__enumext_keyans_store_ref:
__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 ⟨*labels*⟩ and remove the dots "." from them, we do not want to get double dots in references.

```
3280  \cs_new_protected:Nn \__enumext_keyans_store_ref:
3281    {
3282      \bool_if:NT \l__enumext_store_ref_key_bool
3283        {
3284          \cs_set_protected:Npn \__enumext_tmp:n ##1
3285            {
3286              \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
3287              \tl_reverse:c { l__enumext_label_copy_##1_tl }
3288              \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
3289              \tl_reverse:c { l__enumext_label_copy_##1_tl }
3290            }
3291          \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
3292          \__enumext_keyans_store_ref_aux_i:
3293        }
3294    }
```

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

```
3295  \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
3296    {
3297      \bool_if:NT \g__enumext_starred_bool
3298        {
3299          \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
3300        }
3301      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
3302        {
```

```
3303          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
3304            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
3305          }
3306      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
3307        {
3308          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
3309            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
3310          }
3311      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
3312        {
3313          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
3314            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
3315          }
3316      \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
3317        {
3318          \l__enumext_store_name_tl \c_colon_str
3319          \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
3320        }
3321      \__enumext_keyans_store_ref_aux_ii:
3322    }
```

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.

```
3323 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
3324    {
3325      \tl_put_right:Ne \l__enumext_write_aux_file_tl
3326        {
3327          \__enumext_newlabel:nn
3328            { \exp_not:V \l__enumext_newlabel_arg_one_tl }
3329            { \l__enumext_newlabel_arg_two_tl }
3330        }
3331      \l__enumext_write_aux_file_tl
3332    }
```

(*End of definition for* __enumext_keyans_store_ref:, __enumext_keyans_store_ref_aux_i:, *and* __enumext_keyans_-store_ref_aux_ii:.)

13.36.3   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.

```
3333 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
3334    {
3335      \tl_clear:N \l__enumext_store_current_label_tl
3336      \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
3337        {
3338          \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
3339        }
3340        {
3341          \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
3342        }
3343      \tl_if_novalue:nF { #1 }
3344        {
3345          \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_v_tl
3346            {
3347              \tl_put_right:NV \l__enumext_store_current_label_tl \l__enumext_store_keyans_item_opt_
3348            }
3349          \tl_put_right:Nn \l__enumext_store_current_label_tl { #1 }
3350        }
3351      \__enumext_keyans_addto_seq_link:
3352    }
```

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.

```
3353 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
```

```
3354    {
3355      \bool_lazy_and:nnT
3356        { \bool_if_p:N \l__enumext_store_ref_key_bool }
3357        { \bool_if_p:N \l__enumext_hyperref_bool }
3358        {
3359          \tl_put_right:Ne \l__enumext_store_current_label_tl
3360            {
3361              \hfill \exp_not:N \hyperlink
3362                {
3363                  \exp_not:V \l__enumext_newlabel_arg_one_tl
3364                }
3365                { \exp_not:V \l__enumext_mark_ref_sym_tl }
3366            }
3367        }
3368      \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
3369      \bool_if:NT \l__enumext_check_answers_bool
3370        {
3371          \int_gincr:N \g__enumext_item_anskey_int
3372        }
3373    }
```

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

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

__enumext_keyans_save_item_opt:n
__enumext_keyans_show_item_opt:
__enumext_keyans_show_item_opt_viii:

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

```
3374    \cs_new_protected:Npn \__enumext_keyans_save_item_opt:n #1
3375      {
3376        \tl_if_novalue:nF { #1 }
3377          {
3378            \tl_set:Nn \l__enumext_store_current_opt_arg_tl { #1 }
3379          }
3380      }
```

The function __enumext_keyans_show_item_opt: will print the optional arguments of \item* and \anspic* when the show-ans or show-pos keys are set next to the key wrap-opt in keyans and keyanspic environments.

```
3381    \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
3382      {
3383        \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
3384          {
3385            \bool_lazy_or:nnT
3386              { \bool_if_p:N \l__enumext_show_answer_bool }
3387              { \bool_if_p:N \l__enumext_show_position_bool }
3388              {
3389                \__enumext_keyans_wrapper_opt_v:n
3390                  { \l__enumext_store_current_opt_arg_tl } \c_space_tl
3391              }
3392          }
3393      }
```

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

```
3394    \cs_new_protected:Nn \__enumext_keyans_show_item_opt_viii:
3395      {
3396        \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
3397          {
3398            \bool_lazy_or:nnT
3399              { \bool_if_p:N \l__enumext_show_answer_bool }
3400              { \bool_if_p:N \l__enumext_show_position_bool }
3401              {
3402                \__enumext_keyans_wrapper_opt_viii:n
3403                  { \l__enumext_store_current_opt_arg_tl } \c_space_tl
3404              }
3405          }
3406      }
```

(*End of definition for* __enumext_keyans_save_item_opt:n, __enumext_keyans_show_item_opt:, *and* __enumext_keyans_-show_item_opt_viii:.)

__enumext_keyans_pos_mark_set:
__enumext_keyans_show_ans:
__enumext_keyans_show_pos:

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

```
3407 \cs_new_protected:Nn \__enumext_keyans_pos_mark_set:
3408   {
3409     \__enumext_label_width_by_box:Nn
3410       \l__enumext_mark_sep_tmpa_dim { \l__enumext_label_v_tl }
3411     \str_case:Vn \l__enumext_align_label_pos_v_str
3412       {
3413         { l }
3414             {
3415               \dim_set:Nn \l__enumext_mark_sep_tmpb_dim { \c_zero_dim }
3416             }
3417         { r }
3418             {
3419               \dim_set:Nn \l__enumext_mark_sep_tmpb_dim
3420                 { \l__enumext_labelwidth_v_dim - \l__enumext_mark_sep_tmpa_dim }
3421             }
3422         { c }
3423             {
3424               \dim_set:Nn \l__enumext_mark_sep_tmpb_dim
3425                 { 0.5\l__enumext_labelwidth_v_dim - 0.5\l__enumext_mark_sep_tmpa_dim }
3426             }
3427       }
```

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

```
3428     \dim_compare:nNnT { \l__enumext_mark_sym_sep_v_dim } = { \c_zero_dim }
3429       {
3430         \dim_set:Nn \l__enumext_mark_sym_sep_v_dim { \l__enumext_labelsep_v_dim }
3431       }
3432     \tl_set_eq:NN \l__enumext_mark_answer_sym_tl \l__enumext_mark_answer_sym_v_tl
3433     \dim_add:Nn \l__enumext_mark_sym_sep_v_dim { \l__enumext_mark_sep_tmpb_dim }
3434     \str_set_eq:NN \l__enumext_mark_position_str \l__enumext_mark_position_v_str
3435   }
```

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

```
3436 \cs_new_protected:Nn \__enumext_keyans_show_ans:
3437   {
3438     \bool_lazy_all:nT
3439       {
3440         { \bool_if_p:N \l__enumext_show_answer_bool   }
3441         { \bool_if_p:N \l__enumext_item_wrap_key_bool }
3442       }
3443       {
3444         \__enumext_keyans_pos_mark_set:
3445         \__enumext_print_keyans_box:NN
3446           \l__enumext_labelwidth_v_dim \l__enumext_mark_sym_sep_v_dim
3447       }
3448   }
```

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

```
3449 \cs_new_protected:Nn \__enumext_keyans_show_pos:
3450   {
3451     \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
3452       {
3453         \int_incr:N \l__enumext_show_pos_tmp_int
3454       }
3455       {
3456         \int_zero:N \l__enumext_show_pos_tmp_int
3457       }
3458     \bool_lazy_all:nT
3459       {
3460         { \bool_if_p:N \l__enumext_show_position_bool }
3461         { \bool_if_p:N \l__enumext_item_wrap_key_bool }
3462       }
3463       {
3464         \tl_set:Ne \l__enumext_mark_answer_sym_v_tl
3465           {
3466             \group_begin:
3467               \exp_not:N \normalfont
3468               \exp_not:N \footnotesize [ \int_eval:n
```

```
3469                    {
3470                      \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3471                      + \l__enumext_show_pos_tmp_int
3472                    }
3473                  ]
3474                \group_end:
3475            }
3476          \__enumext_keyans_pos_mark_set:
3477          \__enumext_print_keyans_box:NN
3478            \l__enumext_labelwidth_v_dim \l__enumext_mark_sym_sep_v_dim
3479        }
3480    }
```

(*End of definition for* `\__enumext_keyans_pos_mark_set:` *,* `\__enumext_keyans_show_ans:` *, and* `\__enumext_keyans_show_-`
pos: *.*)

13.37 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.

```
3481 \cs_new_protected:Npn \__enumext_default_item:n #1
3482   {
3483     \tl_if_novalue:nTF {#1}
3484       {
3485         \bool_if:NT \l__enumext_check_answers_bool
3486           {
3487             \int_gincr:N \g__enumext_item_number_int
3488             \bool_set_true:N \l__enumext_item_number_bool
3489           }
3490         \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
3491         \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3492       }
3493       {
3494         \bool_set_eq:cc
3495           { l__enumext_wrap_label_ \__enumext_level: _bool }
3496           { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
3497         \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl
3498       }
3499   }
```

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

__enumext_item_starred_exec:nn The \item*, \item*[*symbol*] and \item*[*symbol*][*offset*] works like the *numbered* \item, but
__enumext_item_starred_exec: placing a *symbol* to the "left" of the *label* separated from it by the value the second *optional argument* *offset*.

#1: \l__enumext_item_symbol_X_tl
#2: \l__enumext_item_symbol_sep_X_dim

First we will make a copy of \l__enumext_item_symbol_X_tl which is set by the key item-sym* or passed as "*first*" *optional argument* in the global variable \g__enumext_item_symbol_aux_tl, followed by setting the variable \l__enumext_item_symbol_sep_X_dim set by the key item-pos* or by the "*second*" *optional argument*, then we will see the state of the variable \l__enumext_check_answers_bool set by the key no-store, set the boolean variable \l__enumext_wrap_label_X_bool to "true" for the key wrap-label and execute __enumext_item_std:w and the key itemindent.

```
3500 \cs_new_protected:Npn \__enumext_item_starred_exec:nn #1 #2
3501   {
3502     \tl_if_novalue:nTF {#1}
```

```
3503        {
3504            \tl_gset_eq:Nc
3505                \g__enumext_item_symbol_aux_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
3506        }
3507        {
3508            \tl_gset:Nn \g__enumext_item_symbol_aux_tl {#1}
3509        }
3510    \tl_if_novalue:nTF {#2}
3511        {
3512            \dim_set_eq:cc
3513                { l__enumext_item_symbol_sep_ \__enumext_level: _dim }
3514                { l__enumext_labelsep_ \__enumext_level: _dim }
3515        }
3516        {
3517            \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
3518        }
3519    \bool_if:NT \l__enumext_check_answers_bool
3520        {
3521            \int_gincr:N \g__enumext_item_number_int
3522            \bool_set_true:N \l__enumext_item_number_bool
3523        }
3524    \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
3525    \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3526    }
```

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

```
3527 \cs_new_protected:Nn \__enumext_item_starred_exec:
3528    {
3529        \tl_if_empty:cF { l__enumext_item_symbol_ \__enumext_level: _tl }
3530            {
3531                \mode_leave_vertical:
3532                \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3533                \hbox_overlap_left:n { \g__enumext_item_symbol_aux_tl }
3534                \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3535            }
3536    }
```

(*End of definition for* `\__enumext_item_starred_exec:nn` *and* `\__enumext_item_starred_exec:.`)

`\__enumext_redefine_item:` The function `\__enumext_redefine_item:` will redefine the `\item` command in the enumext environment adding `\item*`. This function are passed to `\__enumext_list_arg_two_X:` used in the definition of the enumext environment (§13.42).

```
3537 \cs_new_protected:Nn \__enumext_redefine_item:
3538    {
3539        \RenewDocumentCommand \item { s o o }
3540            {
3541                \bool_if:nTF {##1}
3542                    {
3543                        \__enumext_item_starred_exec:nn {##2} {##3}
3544                    }
3545                    { \__enumext_default_item:n {##2} }
3546            }
3547    }
```

(*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-
`\__enumext_make_label_std:` label, wrap-label* and `\item*` for enumext environment. This function are passed to `\__enumext_-`
`\__enumext_make_label_box:` `list_arg_two_X:` used in the definition of the enumext environment (§13.42).

```
3548 \cs_new_protected:Nn \__enumext_make_label:
3549    {
3550        \IfDocumentMetadataTF
3551            {
3552                \__enumext_make_label_box:
3553            }
3554            {
3555                \bool_if:NTF \l__enumext_mode_box_bool
3556                    {
3557                        \__enumext_make_label_box:
3558                    }
```

```
3559            {
3560              \__enumext_make_label_std:
3561            }
3562          }
3563      }
```

Standard definition when `\DocumentMetadata` is not active.

```
3564 \cs_new_protected:Nn \__enumext_make_label_std:
3565   {
3566     \RenewDocumentCommand \makelabel { m }
3567       {
3568         \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
3569         \__enumext_item_starred_exec:
3570         \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3571         \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3572           {
3573             \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3574           }
3575           { ##1 }
3576         \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
3577         \tl_gclear:N \g__enumext_item_symbol_aux_tl
3578       }
3579   }
```

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.

```
3580 \cs_new_protected:Nn \__enumext_make_label_box:
3581   {
3582     \RenewDocumentCommand \makelabel { m }
3583       {
3584         \strut\smash
3585           {
3586             \makebox
3587               [ \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim } ]
3588               [ \str_use:c { l__enumext_align_label_pos_ \__enumext_level: _str } ]
3589               {
3590                 \__enumext_item_starred_exec:
3591                 \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3592                 \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3593                   {
3594                     \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3595                   }
3596                   { ##1 }
3597                 \tl_gclear:N \g__enumext_item_symbol_aux_tl
3598               }
3599           } % close smash
3600       }
3601   }
```

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

13.38   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*
```
3602 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
3603   {
3604     \keys_define:nn { enumext / #1 }
3605       {
3606         item-sym* .tl_set:c  = { l__enumext_item_symbol_#2_tl },
3607         item-sym* .value_required:n = true,
3608         item-sym* .initial:n  = {\textborn},
3609         item-pos* .dim_set:c  = { l__enumext_item_symbol_sep_#2_dim },
3610         item-pos* .value_required:n = true,
3611       }
3612   }
3613 \clist_map_inline:nn
3614   {
```

```
3615            {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
3616          }
3617        { \__enumext_tmp:nn #1 }
```

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

13.39 Handling unknown keys

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

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

13.39.1 Handling unknown keys for keyans, keyans* and keyanspic

unknown
__enumext_keyans_unknown_keys:n
__enumext_keyans_unknown_keys:nn

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

```
3618 \cs_set_protected:Npn \__enumext_tmp:n #1
3619   {
3620     \keys_define:nn { enumext / #1 }
3621       {
3622         unknown .code:n = {
3623                         \tl_set:Nn \l__enumext_envir_name_tl {#1}
3624                         \__enumext_keyans_unknown_keys:n {##1}
3625                       },
3626       }
3627   }
3628 \clist_map_inline:nn { keyans, keyans*, keyanspic } { \__enumext_tmp:n {#1} }
```

Internal functions for handling unknown key.

```
3629 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:n #1
3630   {
3631     \exp_args:NV \__enumext_keyans_unknown_keys:nn \l_keys_key_str {#1}
3632   }
3633 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:nn #1#2
3634   {
3635     \tl_if_blank:nTF {#2}
3636       {
3637         \msg_error:nne { enumext } { keyans-unknown-key } {#1}
3638       }
3639       {
3640         \msg_error:nnee { enumext } { keyans-unknown-key-value } {#1} {#2}
3641       }
3642   }
```

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

13.39.2 Handling unknown keys for enumext*

unknown
__enumext_starred_unknown_keys:n
__enumext_starred_unknown_keys:nn

Define and set unknown key for enumext* environment.

```
3643 \keys_define:nn { enumext / enumext* }
3644   {
3645     unknown .code:n = {
3646                     \tl_set:Nn \l__enumext_envir_name_tl { enumext* }
3647                     \__enumext_starred_unknown_keys:n {#1}
3648                   },
3649   }
```

Internal functions for handling unknown key.

```
3650 \cs_new_protected:Npn \__enumext_starred_unknown_keys:n #1
3651   {
3652     \exp_args:NV \__enumext_starred_unknown_keys:nn \l_keys_key_str {#1}
3653   }
3654 \cs_new_protected:Npn \__enumext_starred_unknown_keys:nn #1#2
3655   {
3656     \tl_if_blank:nTF {#2}
3657       {
3658         \msg_error:nne { enumext } { starred-unknown-key } {#1}
3659       }
3660       {
3661         \msg_error:nnee { enumext } { starred-unknown-key-value } {#1} {#2}
3662       }
3663   }
```

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

13.39.3 Handling unknown keys for enumext

unknown
__enumext_standar_unknown_keys:n
__enumext_standar_unknown_keys:nn

Defines and set the key unknown for enumext environment.

```
3664 \cs_set_protected:Npn \__enumext_tmp:n #1
3665   {
3666     \keys_define:nn { enumext / level-#1 }
3667       {
3668         unknown .code:n = {
3669                           \int_set:Nn \l__enumext_level_int { #1 }
3670                           \tl_set:Nn \l__enumext_envir_name_tl { enumext }
3671                           \__enumext_standar_unknown_keys:n {##1}
3672                         },
3673       }
3674   }
3675 \clist_map_inline:nn {1, 2, 3, 4} { \__enumext_tmp:n {#1} }
```

Internal functions for handling unknown key.

```
3676 \cs_new_protected:Npn \__enumext_standar_unknown_keys:n #1
3677   {
3678     \exp_args:NV \__enumext_standar_unknown_keys:nn \l_keys_key_str {#1}
3679   }
3680 \cs_new_protected:Npn \__enumext_standar_unknown_keys:nn #1#2
3681   {
3682     \tl_if_blank:nTF {#2}
3683       {
3684         \msg_error:nne { enumext } { standar-unknown-key } {#1}
3685       }
3686       {
3687         \msg_error:nnee { enumext } { standar-unknown-key-value } {#1} {#2}
3688       }
3689   }
```

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

13.40 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.

```
3690 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
3691   {
3692     \tl_if_novalue:nTF { #1 }
3693       {
3694         \bool_set_true:N \l__enumext_wrap_label_v_bool
3695         \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
3696       }
3697       {
3698         \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
3699         \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
3700       }
3701   }
```

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

__enumext_keyans_starred_item:n

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

```
3702 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
3703   {
3704     \__enumext_keyans_save_item_opt:n { #1 }
3705     \bool_set_true:N \l__enumext_wrap_label_v_bool
3706     \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
3707     \__enumext_keyans_show_item_opt:
```

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

```
3708     \__enumext_keyans_addto_prop:n { #1 }
```

```
3709        \__enumext_keyans_store_ref:
3710        \__enumext_keyans_addto_seq:n { #1 }
3711        \int_gincr:N \g__enumext_check_starred_cmd_int
3712    }
```

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

\item*
__enumext_keyans_redefine_item:

The function __enumext_keyans_redefine_item: is responsible for adding the *starred argument* and *optional argument* by the __enumext_list_arg_two_v: function in the definition of the keyans environment. Here we will set to true the variable \l__enumext_item_wrap_key_bool used by the wrap-ans* key only when \item* is executed and additionally we need to use \peek_remove_spaces:n to avoid an unwanted space when using \item* together with the itemindent key. This function are passed to __enumext_list_arg_two_v: used in the definition of the keyans environment (§13.41).

```
3713  \cs_new_protected:Nn \__enumext_keyans_redefine_item:
3714    {
3715        \RenewDocumentCommand \item { s o }
3716          {
3717            \bool_if:nTF {##1}
3718              {
3719                \bool_set_true:N \l__enumext_item_wrap_key_bool % wrap-ans*
3720                \peek_remove_spaces:n
3721                  {
3722                    \__enumext_keyans_starred_item:n {##2}
3723                  }
3724              }
3725              {
3726                \bool_set_false:N \l__enumext_item_wrap_key_bool
3727                \__enumext_keyans_default_item:n {##2}
3728              }
3729          }
3730    }
```

(*End of definition for* \item* *and* __enumext_keyans_redefine_item:. *This function is documented on page* 17.)

__enumext_keyans_make_label:
__enumext_keyans_wrapper_label:n
__enumext_keyans_make_label_std:
__enumext_keyans_make_label_box:

The function __enumext_keyans_make_label: redefine \makelabel for the keys mode-box, align, font, wrap-label, wrap-label*, wrap-ans* and \item* for keyans environment. This function are passed to __enumext_list_arg_two_v: used in the definition of the keyans environment (§13.41).

```
3731  \cs_new_protected:Nn \__enumext_keyans_make_label:
3732    {
3733        \IfDocumentMetadataTF
3734          {
3735            \__enumext_keyans_make_label_box:
3736          }
3737          {
3738            \bool_if:NTF \l__enumext_mode_box_bool
3739              {
3740                \__enumext_keyans_make_label_box:
3741              }
3742              {
3743                \__enumext_keyans_make_label_std:
3744              }
3745          }
3746    }
```

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

```
3747  \cs_new_protected:Npn \__enumext_keyans_wrapper_label:n #1
3748    {
3749        \bool_lazy_all:nT
3750          {
3751            { \bool_if_p:N \l__enumext_wrap_label_v_bool        }
3752            { \bool_if_p:N \l__enumext_show_answer_bool         }
3753            { \bool_if_p:N \l__enumext_item_wrap_key_bool       }
3754            { \cs_if_exist_p:N \__enumext_keyans_wrapper_item_v:n }
3755          }
3756          {
3757            \cs_set_eq:NN \__enumext_wrapper_label_v:n \__enumext_keyans_wrapper_item_v:n
3758          }
3759        \bool_if:NTF \l__enumext_wrap_label_v_bool
3760          {
3761            \__enumext_wrapper_label_v:n { #1 }
```

```
3762          }
3763        { #1 }
3764    }
```

Standard definition when `\DocumentMetadata` is not active.

```
3765  \cs_new_protected:Nn \__enumext_keyans_make_label_std:
3766    {
3767      \RenewDocumentCommand \makelabel { m }
3768        {
3769          \tl_use:N \l__enumext_label_fill_left_v_tl
3770          \__enumext_keyans_show_ans:
3771          \__enumext_keyans_show_pos:
3772          \tl_use:N \l__enumext_label_font_style_v_tl
3773          \__enumext_keyans_wrapper_label:n { ##1 }
3774          \tl_use:N \l__enumext_label_fill_right_v_tl
3775        }
3776    }
```

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

```
3777  \cs_new_protected:Nn \__enumext_keyans_make_label_box:
3778    {
3779      \RenewDocumentCommand \makelabel { m }
3780        {
3781          \strut\smash
3782            {
3783              \makebox[ \l__enumext_labelwidth_v_dim ][ \l__enumext_align_label_pos_v_str ]
3784                {
3785                  \__enumext_keyans_show_ans:
3786                  \__enumext_keyans_show_pos:
3787                  \tl_use:N \l__enumext_label_font_style_v_tl
3788                  \__enumext_keyans_wrapper_label:n { ##1 }
3789                }
3790            }
3791        }
3792    }
```

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

13.41   Second argument of the lists

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

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

13.41.1   Calculation of `\leftmargin` and `\itemindent`

Consider the figure 9 where the default margins (on the left) of a list are represented.
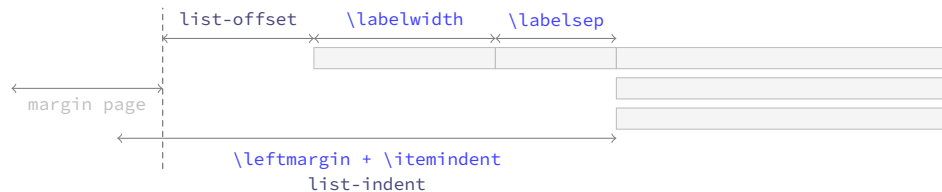
Figure 9: Representation of standard horizontal lengths in `list` environment.

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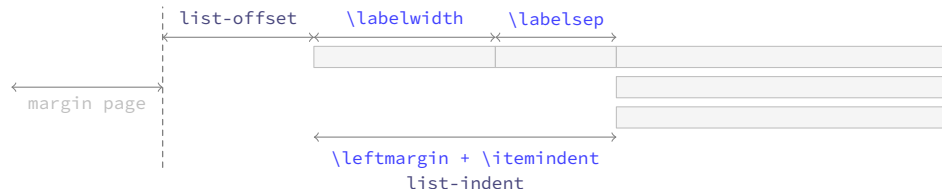
Figure 10: Representation of horizontal lengths concept in list in enumext.

Where the default values will look like in the figure 11.
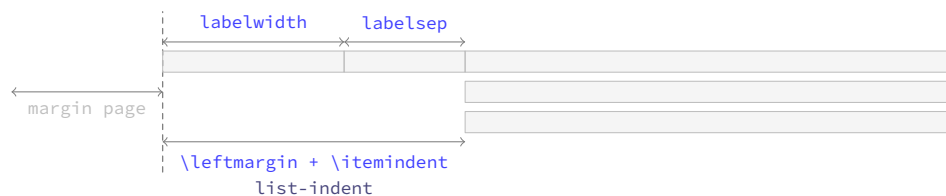
Figure 11: Default horizontal lengths in enumext.

__enumext_calc_hspace:NNNNNNN
__enumext_calc_hspace:ccccccc

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

#1: \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

And returns the *"adjusted"* values of \leftmargin and \itemindent.

```
3793 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
3794   {
3795     \dim_compare:nNnT { #1 } < { \c_zero_dim }
3796       {
3797         \msg_warning:nnnV { enumext } { width-non-positive }{ labelwidth } #1
3798         \dim_set:Nn #1 { \dim_abs:n { #1 } }
3799       }
3800     \dim_compare:nNnT { #2 } < { \c_zero_dim }
3801       {
3802         \msg_warning:nnnV { enumext } { width-negative }{ labelsep }  #2
3803         \dim_set:Nn #2 { \dim_abs:n { #2 } }
3804       }
```

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

```
3805       \bool_if:NF #7 { \dim_set:Nn #4 { #1 + #2} }
```

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

```
3806       \dim_compare:nNnTF { #4 } < { \c_zero_dim }
3807         {
3808         \dim_set:Nn #6 { #1 + #2 - #4}
3809         \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3810         }
3811         {
3812         \dim_compare:nNnT { #4 } = { #1 + #2 }
3813           { \dim_set:Nn #6 { \c_zero_dim } }
3814         \dim_compare:nNnT { #4 } < { #1 + #2 }
3815           { \dim_set:Nn #6 { #1 + #2 - #4} }
3816         \dim_compare:nNnT { #4 } > { #1 + #2 }
3817           {
3818           \dim_set:Nn #6 { -#1 - #2 + #4}
3819           \dim_set:Nn #6 { #6*-1}
3820           }
3821         \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3822         }
3823   }
3824 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { ccccccc }
```

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

13.41.2 Setting second argument of the lists

__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:

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.

```
3825 \cs_set_protected:Npn \__enumext_tmp:n #1
3826   {
3827     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3828       {
3829         \__enumext_calc_hspace:ccccccc
3830           { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3831           { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3832           { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3833           { l__enumext_leftmargin_tmp_#1_bool }
```

```
3834          \clist_map_inline:nn
3835            { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3836            { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
3837          \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3838            { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
3839          \clist_map_inline:nn { beginparpenalty, itempenalty, endparpenalty }
3840            { \int_set_eq:cc {@####1} { l__enumext_####1_#1_int } }
3841          \usecounter { enumX#1 }
3842          \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
3843          \str_if_eq:nnTF {#1} { v }
3844            {
3845              \__enumext_keyans_redefine_item:
3846              \__enumext_keyans_make_label:
3847              \__enumext_keyans_ref:
3848              \__enumext_keyans_fake_item_indent:
3849              \bool_if:cT { l__enumext_show_length_#1_bool }
3850                {
3851                  \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3852                }
3853            }
3854            {
3855              \__enumext_redefine_item:
3856              \__enumext_make_label:
3857              \__enumext_standar_ref:
3858              \__enumext_fake_item_indent:
3859              \bool_if:cT { l__enumext_show_length_#1_bool }
3860                {
3861                  \msg_term:nnne { enumext } { list-lengths } {#1}
3862                    { \int_use:N \l__enumext_level_int }
3863                }
3864            }
3865        }
3866      }
3867    \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
`\__enumext_list_arg_two_viii:`   `\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.

```
3868    \cs_set_protected:Npn \__enumext_tmp:n #1
3869      {
3870        \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3871          {
3872            \bool_set_true:c { l__enumext_leftmargin_tmp_#1_bool }
3873            \dim_zero:c { l__enumext_leftmargin_tmp_#1_dim }
3874            \__enumext_calc_hspace:ccccccc
3875              { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3876              { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3877              { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3878              { l__enumext_leftmargin_tmp_#1_bool }
3879            \clist_map_inline:nn
3880              { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3881              { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
3882            \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3883              { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
3884            \clist_map_inline:nn { beginparpenalty, itempenalty, endparpenalty }
3885              { \int_set_eq:cc {@####1} { l__enumext_####1_#1_int } }
3886            \skip_set_eq:Nc \parsep  { l__enumext_itemsep_#1_skip }
3887            \skip_zero:N \partopsep
3888            \usecounter { enumX#1 }
3889            \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
3890            \__enumext_starred_ref:
3891            \str_if_eq:nnTF {#1} { vii }
3892              {
3893                \__enumext_fake_item_indent_vii:
3894                \bool_if:cT { l__enumext_show_length_vii_bool }
3895                  { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3896              }
3897              {
```

```
3898            \__enumext_fake_item_indent_viii:
3899              \bool_if:cT { l__enumext_show_length_#1_bool }
3900                { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3901              }
3902          }
3903      }
3904  \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:.)

13.42   The environment enumext

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

```
3905  \cs_new_protected:Nn \__enumext_safe_exec:
3906    {
3907      \__enumext_is_not_nested:
3908      \__enumext_internal_mini_page:
3909      \int_incr:N \l__enumext_level_int
3910      \int_compare:nNnT { \l__enumext_level_int } > { 4 }
3911        { \msg_fatal:nn { enumext } { list-too-deep } }
3912      \bool_set_true:N \l__enumext_standar_bool
3913      \bool_set_false:N \l__enumext_starred_bool
3914      \__enumext_is_on_first_level:
3915    }
```

(*End of definition for* __enumext_safe_exec:.)

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

```
3916  \cs_new_protected:Npn \__enumext_parse_keys:n #1
3917    {
3918      \tl_if_novalue:nF {#1}
3919        {
3920          \str_clear:N \l__enumext_series_name_str
3921          \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
3922            {
3923              \keys_set:nn { enumext / level-1 } {#1}
3924              \bool_if:NF \l__enumext_print_keyans_cmd_bool
3925                {
3926                  \__enumext_parse_series:n {#1}
3927                }
3928              \__enumext_nested_base_line_fix:
3929            }
3930            {
3931              \exp_args:Ne \keys_set:nn
3932                { enumext / level-\int_use:N \l__enumext_level_int } {#1}
3933              \bool_if:NF \l__enumext_print_keyans_cmd_bool
3934                {
3935                  \__enumext_parse_series:n {#1}
3936                }
3937            }
3938          \__enumext_store_active_keys:n {#1}
3939        }
3940    }
```

(*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*.

```
3941  \cs_new_protected:Nn \__enumext_start_store_level:
3942    {
3943      \bool_lazy_all:nT
```

```
3944            {
3945                { \bool_if_p:N \l__enumext_store_active_bool }
3946                { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3947                { \bool_if_p:N \g__enumext_standar_bool }
3948            }
3949            {
3950                \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3951                  {
3952                    \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3953                    \__enumext_store_level_open:
3954                  }
3955            }
```

If enumext are nested in enumext* add __enumext_store_level_open: to preserve the *"storing structure"*.

```
3956        \bool_lazy_all:nT
3957            {
3958                { \bool_if_p:N \l__enumext_store_active_bool }
3959                { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3960                { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
3961            }
3962            {
3963                \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3964                  {
3965                    \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3966                    \__enumext_store_level_open:
3967                  }
3968            }
3969    }
```

(*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*.

```
3970 \cs_new_protected:Nn \__enumext_stop_store_level:
3971    {
3972        \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
3973            {
3974                \__enumext_store_level_close:
3975            }
3976    }
```

(*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=0pt and set the value of \multicolsep equal to zero and leave \columnseprule equal to zero for inner levels.

```
3977 \cs_new_protected:Nn \__enumext_multicols_start:
3978    {
3979        \int_compare:nNnT
3980            { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3981            {
3982                \dim_compare:nNnT
3983                    { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3984                    {
3985                        \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
3986                          {
3987                            ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
3988                              + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
3989                            ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
3990                            - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3991                          }
3992                    }
3993                \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
3994                \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3995                    {
3996                        \dim_zero:N \columnseprule
3997                    }
```

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.

```
3998                \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3999                    {
```

```
4000              \skip_zero:N \multicolsep
4001              \__enumext_multi_addvspace:
4002            }
4003        \raggedcolumns
4004        \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
4005      }
4006  }
```

(*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:.

```
4007 \cs_new_protected:Nn \__enumext_multicols_stop:
4008  {
4009    \int_compare:nNnTF
4010      { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
4011      {
4012        \__enumext_stop_list:
4013        \__enumext_stop_store_level:
4014        \end{multicols}
4015        \__enumext_unskip_unkern:
4016        \__enumext_unskip_unkern:
4017        \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
4018      }
4019      {
4020        \__enumext_stop_list:
4021        \__enumext_stop_store_level:
4022      }
4023  }
```

(*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.

```
4024 \cs_new_protected:Nn \__enumext_before_list:
4025  {
4026    \__enumext_vspace_above:
4027    \__enumext_before_args_exec:
4028    \__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.

```
4029      \dim_compare:nNnT
4030        { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
4031        {
4032          \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
4033            {
4034              \linewidth
4035              - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
4036              - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
4037            }
```

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

```
4038          \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
4039          \int_gincr:N \g__enumext_minipage_stat_int
4040          \__enumext_minipage_add_space:
4041          \noindent
4042          \__enumext_mini_page{ \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
4043        }
4044      \__enumext_multicols_start:
4045  }
```

(*End of definition for* __enumext_before_list:.)

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

```
4046 \cs_new_protected:Nn \__enumext_second_part:
4047   {
4048     \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
4049       {
4050         \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
4051           {
4052             \msg_warning:nn { enumext } { missing-miniright }
4053             \miniright
4054           }
4055         \int_gzero:N \g__enumext_minipage_stat_int
4056         \__enumext_unskip_unkern: % remove topsep + [partopsep]
4057         \end__enumext_mini_page
4058       }
4059       {
4060         \__enumext_multicols_stop:
4061       }
```

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.

```
4062     \__enumext_after_stop_list:
4063     \__enumext_check_ans_key_hook:
4064     \__enumext_vspace_below:
4065     \bool_set_false:N \l__enumext_standar_bool
4066     \bool_if:NF \l__enumext_print_keyans_cmd_bool
4067       {
4068         \__enumext_standar_save_counter:
4069       }
4070   }
```

(*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.

```
4071 \cs_new_protected:Nn \__enumext_set_item_width:
4072   {
4073     \dim_set:Nn \itemwidth { \linewidth }
4074     \dim_compare:nT
4075       {
4076         \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim } != \c_zero_dim
4077       }
4078       {
4079         \dim_sub:Nn \itemwidth
4080           {
4081             \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
4082           }
4083       }
4084   }
```

(*End of definition for* __enumext_set_item_width:.)

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

```
4085 \NewDocumentEnvironment{enumext}{ O{} }
4086   {
4087     \__enumext_safe_exec:
4088     \__enumext_parse_keys:n {#1}
4089     \__enumext_before_list:
4090     \__enumext_start_store_level:
4091     \__enumext_start_list:nn
4092       { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
4093       {
4094         \use:c { __enumext_list_arg_two_ \__enumext_level: : }
4095         \__enumext_before_keys_exec:
```

```
4096        }
4097      \__enumext_set_item_width:
4098      \__enumext_after_args_exec:
4099    }
4100    {
4101      \__enumext_second_part:
4102    }
```

(*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.

```
4103  \__enumext_after_env:nn {enumext}
4104    {
4105      \__enumext_execute_after_env:
4106    }
```

13.43   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"*.

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

```
4107  \cs_new_protected:Nn \__enumext_keyans_safe_exec:
4108    {
4109      \bool_if:NF \l__enumext_store_active_bool
4110        {
4111          \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
4112        }
4113      \int_incr:N \l__enumext_keyans_level_int
4114      \bool_set_true:N \l__enumext_keyans_env_bool
4115      \__enumext_keyans_name_and_start:
4116      % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
4117      \bool_set_false:N \l__enumext_store_active_bool
4118      \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
4119        {
4120          \msg_error:nn { enumext } { keyans-nested }
4121        }
4122      \int_compare:nNnT { \l__enumext_level_int } > { 1 }
4123        {
4124          \msg_error:nn { enumext } { keyans-wrong-level }
4125        }
4126    }
```

(*End of definition for* __enumext_keyans_safe_exec:.)

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

```
4127  \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
4128    {
4129      \keys_set:nn { enumext / keyans } {#1}
4130    }
```

(*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:

```
4131  \cs_new_protected:Nn \__enumext_before_list_v:
4132    {
4133      \__enumext_vspace_above_v:
4134      \__enumext_before_args_exec_v:
4135      \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
4136        {
4137          \dim_set:Nn \l__enumext_minipage_left_v_dim
4138            {
4139              \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
4140            }
4141          \bool_set_true:N \l__enumext_minipage_active_v_bool
4142          \int_gincr:N \g__enumext_minipage_stat_int
4143          \__enumext_keyans_minipage_add_space:
4144          \__enumext_mini_page{ \l__enumext_minipage_left_v_dim }
```

```
4145            }
4146          \__enumext_keyans_multicols_start:
4147      }
4148  \cs_new_protected:Nn \__enumext_keyans_multicols_start:
4149      {
4150        \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
4151          {
4152            \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
4153              {
4154                \dim_set:Nn \l__enumext_columns_sep_v_dim
4155                  {
4156                    (
4157                      \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
4158                    ) / \l__enumext_columns_v_int
4159                    - \l__enumext_listoffset_v_dim
4160                  }
4161              }
4162            \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim
4163            \dim_zero:N \columnseprule % no rule here
4164            \bool_if:NF \l__enumext_minipage_active_v_bool
4165              {
4166                \skip_zero:N \multicolsep
4167                \__enumext_keyans_multi_addvspace:
4168              }
4169            \raggedcolumns
4170            \begin{multicols}{ \l__enumext_columns_v_int }
4171          }
4172      }
4173  \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
4174      {
4175        \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
4176          {
4177            \__enumext_stop_list:
4178            \end{multicols}
4179            \__enumext_unskip_unkern:
4180            \__enumext_unskip_unkern:
4181            \par\addvspace{ \l__enumext_multicols_below_v_skip }
4182          }
4183          {
4184            \__enumext_stop_list:
4185          }
4186      }
4187  \cs_new_protected:Nn \__enumext_second_part_v:
4188      {
4189        \bool_if:NTF \l__enumext_minipage_active_v_bool
4190          {
4191            \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
4192              {
4193                \msg_warning:nn { enumext } { missing-miniright }
4194                \miniright
4195              }
4196            \int_gzero:N \g__enumext_minipage_stat_int
4197            \__enumext_unskip_unkern: % remove \topsep + [\partopsep]
4198            \end__enumext_mini_page
4199            \par\addvspace{ \l__enumext_minipage_after_skip }
4200          }
4201          {
4202            \__enumext_keyans_multicols_stop:
4203          }
4204        \bool_set_false:N \l__enumext_keyans_env_bool
4205        \__enumext_after_stop_list_v:
4206        \__enumext_vspace_below_v:
4207      }
```

(*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.

```
4208  \cs_new_protected:Nn \__enumext_keyans_set_item_width:
4209      {
4210        \dim_set:Nn \itemwidth { \linewidth }
```

```
4211    \dim_compare:nT
4212      {
4213        \l__enumext_listoffset_v_dim != \c_zero_dim
4214      }
4215      {
4216        \dim_sub:Nn \itemwidth { \l__enumext_listoffset_v_dim }
4217      }
4218  }
```

(*End of definition for* __*enumext_keyans_set_item_width:*.)

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

```
4219  \NewDocumentEnvironment{keyans}{ O{} }
4220    {
4221      \__enumext_keyans_safe_exec:
4222      \__enumext_keyans_parse_keys:n {#1}
4223      \__enumext_before_list_v:
4224      \__enumext_start_list:nn
4225        { \tl_use:N \l__enumext_label_v_tl }
4226        {
4227          \__enumext_list_arg_two_v:
4228          \__enumext_before_keys_exec_v:
4229        }
4230      \__enumext_keyans_set_item_width:
4231      \__enumext_after_args_exec_v:
4232    }
4233    {
4234      \__enumext_check_starred_cmd:n { item }
4235      \__enumext_second_part_v:
4236    }
```

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

13.44   Tagging PDF support for non-standart list environments

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

To maintain a correct list structure when \DocumentMetadata is active, it is necessary to do some things manually using tagpdf[18] and ltsockets[20]. This implementation is an adaptation of my answer thanks to Ulrike Fischer's comments in How can I modify my \item redefinition to be compatible with tagging-pdf.

13.44.1   Socket for tagging support in enumext* and keyans*

start-list-tags   We will first define the necessary sockets and their behavior for enumext* and keyans*.
stop-start-tags
stop-list-tags
__enumext_start_list_tag:n
__enumext_stop_start_list_tag:
__enumext_stop_list_tag:n

```
4237  \socket_new:nn {tagsupport/__enumext/starred}{ 1 }
4238  \socket_new_plug:nnn {tagsupport/__enumext/starred} {start-list-tags}
4239    {
4240      \tag_resume:n {#1}
4241      \tag_mc_end_push:
4242        \tag_struct_begin:n {tag=LI}
4243          \tag_struct_begin:n {tag=Lbl}
4244            \tag_mc_begin:n {tag=Lbl}
4245    }
4246  \socket_new_plug:nnn {tagsupport/__enumext/starred} {stop-start-tags}
4247    {
4248            \tag_mc_end:
4249          \tag_struct_end:n {tag=Lbl}
4250          \tag_struct_begin:n {tag=LBody}
4251            \tag_struct_begin:n {tag=text-unit}
4252              \tag_struct_begin:n {tag=text}
4253    }
4254  \socket_new_plug:nnn {tagsupport/__enumext/starred} {stop-list-tags}
4255    {
4256              \tag_struct_end:n {tag=text}
4257            \tag_struct_end:n {tag=text-unit}
4258          \tag_struct_end:n {tag=LBody}
4259        \tag_struct_end:n {tag=LI}
4260      \tag_mc_begin_pop:n {}
4261      \tag_suspend:n {#1}
4262    }
```

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

```
4263 \cs_new_protected_nopar:Npn \__enumext_start_list_tag:n #1
4264   {
4265     \IfDocumentMetadataT
4266       {
4267         \socket_assign_plug:nn {tagsupport/__enumext/starred} {start-list-tags}
4268         \socket_use:nn {tagsupport/__enumext/starred} {#1}
4269       }
4270   }
4271 \cs_new_protected_nopar:Nn \__enumext_stop_start_list_tag:
4272   {
4273     \IfDocumentMetadataT
4274       {
4275         \socket_assign_plug:nn {tagsupport/__enumext/starred} {stop-start-tags}
4276         \socket_use:nn {tagsupport/__enumext/starred} { }
4277       }
4278   }
4279 \cs_new_protected_nopar:Npn \__enumext_stop_list_tag:n #1
4280   {
4281     \IfDocumentMetadataT
4282       {
4283         \socket_assign_plug:nn {tagsupport/__enumext/starred} {stop-list-tags}
4284         \socket_use:nn {tagsupport/__enumext/starred} {#1}
4285       }
4286   }
```

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

13.44.2 Socket for tagging support in `keyanspic`

start-list-tags
stop-start-tags
stop-list-tags
__enumext_anspic_start_list_tag:
__enumext_anspic_stop_start_list_tag:
__enumext_anspic_stop_list_tag:

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

```
4287 \socket_new:nn {tagsupport/__enumext/keyanspic}{ 0 }
4288 \socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {start-list-tags}
4289   {
4290     \tag_resume:n {keyanspic}
4291     \tag_mc_end_push:
4292       \tag_struct_begin:n {tag=LI}
4293         \tag_struct_begin:n {tag=Lbl}
4294           \tag_mc_begin:n {tag=Lbl}
4295   }
4296 \socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {stop-start-tags}
4297   {
4298             \tag_mc_end:
4299         \tag_struct_end:n {tag=Lbl}
4300         \tag_struct_begin:n {tag=LBody}
4301           \tag_struct_begin:n {tag=text-unit}
4302             \tag_struct_begin:n {tag=text}
4303               \tag_mc_begin:n {tag=text}
4304   }
4305 \socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {stop-list-tags}
4306   {
4307               \tag_mc_end:
4308             \tag_struct_end:n {tag=text}
4309           \tag_struct_end:n {tag=text-unit}
4310         \tag_struct_end:n {tag=LBody}
4311       \tag_struct_end:n {tag=LI}
4312     \tag_mc_begin_pop:n {}
4313     \tag_suspend:n {keyanspic}
4314   }
```

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

```
4315 \cs_new_protected_nopar:Nn \__enumext_anspic_start_list_tag:
4316   {
4317     \IfDocumentMetadataT
4318       {
4319         \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {start-list-tags}
4320         \socket_use:n {tagsupport/__enumext/keyanspic}
4321       }
4322   }
4323 \cs_new_protected_nopar:Nn \__enumext_anspic_stop_start_list_tag:
4324   {
4325     \IfDocumentMetadataT
```

```
4326              {
4327                  \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {stop-start-tags}
4328                  \socket_use:n {tagsupport/__enumext/keyanspic}
4329              }
4330          }
4331  \cs_new_protected_nopar:Nn \__enumext_anspic_stop_list_tag:
4332      {
4333          \IfDocumentMetadataT
4334              {
4335                  \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {stop-list-tags}
4336                  \socket_use:n {tagsupport/__enumext/keyanspic}
4337              }
4338          }
```

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

13.45 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.
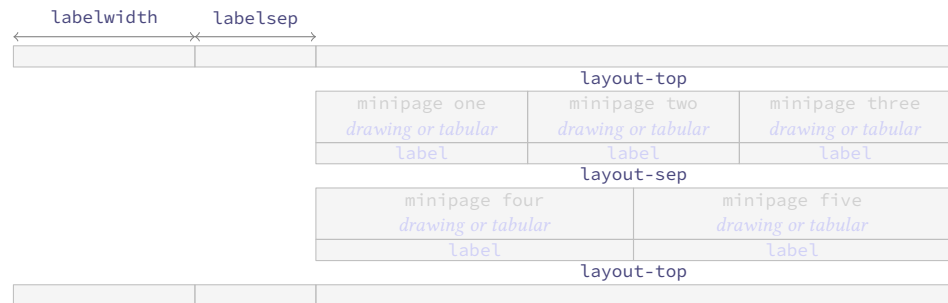
Figure 12: Representation of the `keyanspic` spacing in enumext.

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?.

13.45.1 The environment `keyanspic`

label-pos
label-sep
layout-sty
layout-sep
layout-top
mark-ans
mark-pos
mark-sep
save-sep
wrap-opt
wrap-ans*
show-ans
show-pos

First we define the keys 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`.

```
4339  \keys_define:nn { enumext / keyanspic }
4340      {
4341          label-pos .choice:,
4342          label-pos / above   .code:n =
4343                              \bool_set_true:N \l__enumext_anspic_label_above_bool
4344                              \str_set:Nn \l__enumext_anspic_mini_pos_str { t },
4345          label-pos / below   .code:n =
4346                              \bool_set_false:N \l__enumext_anspic_label_above_bool
4347                              \str_set:Nn \l__enumext_anspic_mini_pos_str { b },
4348          label-pos / unknown .code:n =
4349                              \msg_error:nneee { enumext } { unknown-choice }
4350                                { label-pos } { above,~ below } { \exp_not:n {#1} },
4351          label-pos  .initial:n       = below,
4352          label-pos  .value_required:n = true,
4353          label-sep  .skip_set:N       = \l__enumext_anspic_label_sep_skip,
4354          label-sep  .value_required:n = true,
4355          layout-sty .tl_set:N         = \l__enumext_anspic_layout_style_tl,
4356          layout-sty .value_required:n = true,
4357          layout-sep .code:n           = \keys_set:nn { enumext / keyans } { parsep = #1 },
4358          layout-sep .value_required:n = true,
4359          layout-top .code:n           = \keys_set:nn { enumext / keyans } { topsep = #1 },
4360          layout-top .value_required:n = true,
```

```
4361       mark-ans   .code:n             = \keys_set:nn { enumext / keyans } { mark-ans = #1 },
4362       mark-ans   .value_required:n = true,
4363       mark-pos   .code:n             = \keys_set:nn { enumext / keyans } { mark-pos = #1 },
4364       mark-pos   .value_required:n = true,
4365       mark-sep   .code:n             = \keys_set:nn { enumext / keyans } { mark-sep = #1 },
4366       mark-sep   .value_required:n = true,
4367       save-sep   .code:n             = \keys_set:nn { enumext / keyans } { save-sep = #1 },
4368       save-sep   .value_required:n = true,
4369       wrap-opt   .code:n             = \keys_set:nn { enumext / keyans } { wrap-opt = #1 },
4370       wrap-opt   .value_required:n = true,
4371       wrap-ans*  .code:n             = \keys_set:nn { enumext / keyans } { wrap-ans* = #1 },
4372       wrap-ans*  .value_required:n = true,
4373       show-ans   .code:n             = \keys_set:nn { enumext / keyans } { show-ans = #1 },
4374       show-ans   .value_required:n = true,
4375       show-pos   .code:n             = \keys_set:nn { enumext / keyans } { show-pos = #1 },
4376       show-pos   .value_required:n = true,
4377       unknown    .code:n             = {
4378                                         \tl_set:Nn \l__enumext_envir_name_tl { keyanspic }
4379                                         \__enumext_keyans_unknown_keys:n {#1}
4380                                       },
4381     }
```

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

`\__enumext_keyans_pic_safe_exec:`
`\__enumext_keyans_pic_parse_keys:n`
`\__enumext_keyans_pic_skip_abs:N`
`\__enumext_keyans_pic_arg_two:`

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

```
4382 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
4383   {
4384     \int_incr:N \l__enumext_keyans_pic_level_int
4385     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
4386       {
4387         \msg_error:nn { enumext } { keyanspic-nested }
4388       }
4389     \__enumext_keyans_name_and_start:
4390   }
```

Parse [⟨*key* = *val*⟩] for keyanspic environment.

```
4391 \cs_new_protected:Npn \__enumext_keyans_pic_parse_keys:n #1
4392   {
4393     \tl_if_novalue:nF {#1}
4394       {
4395         \keys_set:nn { enumext / keyanspic } {#1}
4396       }
4397   }
```

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

```
4398 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
4399   {
4400     \dim_compare:nNnT { #1 } < { \c_zero_dim }
4401       {
4402         \skip_set:Nn #1 { -#1 }
4403       }
4404   }
```

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.

```
4405 \cs_new_protected:Npn \__enumext_keyans_pic_arg_two:
4406   {
4407     \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
4408     \__enumext_list_arg_two_v:
4409     \__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*.

```
4410       \bool_if:NF \l__enumext_anspic_label_above_bool
4411         {
```

```
4412          \stepcounter { enumXv }
4413          \hbox_set:Nn \l__enumext_anspic_label_box { \l__enumext_label_v_tl }
4414          \dim_set:Nn \l__enumext_anspic_label_htdp_dim
4415            {
4416              \box_ht_plus_dp:N \l__enumext_anspic_label_box
4417            }
4418          \skip_add:Nn \parsep
4419            {
4420              \l__enumext_anspic_label_htdp_dim
4421              + \box_dp:N \strutbox
4422              + \l__enumext_anspic_label_sep_skip
4423            }
4424        }
```

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.

```
4425        \dim_add:Nn \leftmargin { -\l__enumext_labelwidth_v_dim - \l__enumext_labelsep_v_dim }
4426        \ignorespaces
4427        \skip_add:Nn \topsep { 0.5\box_dp:N \strutbox }
4428        \dim_zero:N  \listparindent
4429        \skip_zero:N \partopsep
4430        \skip_zero:N \itemsep
4431      }
```

(*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.

```
4432    \NewDocumentEnvironment{keyanspic}{ o }
4433      {
4434        \__enumext_keyans_pic_safe_exec:
4435        \__enumext_keyans_pic_parse_keys:n {#1}
4436        \begin{list} { } { \__enumext_keyans_pic_arg_two: }
4437        \IfDocumentMetadataT
4438          {
4439            \tag_suspend:n {list}
4440          }
4441        \item[] \scan_stop:
4442        \RenewDocumentCommand \item {}
4443          {
4444            \msg_error:nn { enumext } { keyanspic-item-cmd }
4445          }
4446        \IfDocumentMetadataT
4447          {
4448            \tag_resume:n {keyanspic}
4449            \tag_tool:n {para/tagging=false}
4450            \tag_suspend:n {keyanspic}
4451          }
4452      }
```

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.

```
4453      {
4454        \IfDocumentMetadataT
4455          {
4456            \tag_resume:n {keyanspic}
4457            \tag_mc_end_push:
4458            \tag_struct_begin:n {tag=L,attribute=enumerate}
4459          }
4460        \__enumext_anspic_exec:
4461        \IfDocumentMetadataT
4462          {
4463            \tag_suspend:n {keyanspic}
4464          }
4465        \end{list}
4466        \IfDocumentMetadataT
4467          {
4468            \tag_struct_end:n {tag=L}
4469            \tag_mc_begin_pop:n {}
4470            \tag_struct_end:n {tag=L}
```

```
4471                \tag_mc_begin_pop:n {}
4472            }
```

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

```
4473        \__enumext_check_starred_cmd:n { anspic }
4474        \setcounter { enumXvi } { 0 }
4475        \bool_if:NTF \l__enumext_anspic_label_above_bool
4476            {
4477                \par\addvspace{ 0.5\box_dp:N \strutbox }
4478            }
4479            {
4480                \par
4481                \addvspace
4482                    {
4483                        \dim_eval:n
4484                            {
4485                                \l__enumext_anspic_label_htdp_dim + \box_ht_plus_dp:N \strutbox
4486                                + \l__enumext_anspic_label_sep_skip + \l__enumext_topsep_v_skip
4487                            }
4488                    }
4489            }
4490        }
```

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

13.45.2   The command `\anspic`

The `\anspic` command take three arguments, the *starred versions* `\anspic*`[⟨*content*⟩] *store* the current ⟨*label*⟩ next to the *optional argument* [⟨*content*⟩] in the *sequence* and *prop list* defined by save-ans key. The third *mandatory argument* {⟨*drawing or tabular*⟩} is NOT *stored* in the *sequence* or *prop list*.

🔷 One of the complications here to make the keyanspic environment compatible with *tagged* PDF is the position of ⟨*label*⟩, the `\anspic` command processes the arguments in order, where #1 and #2 correspond to ⟨*label*⟩ and #3 to the mandatory argument and puts all this inside a minipage environment. If #1 and #2, that is ⟨*label*⟩, is above #3 there are no problems with *tagged* PDF, but if #3 comes first the list created with *tagged* PDF will not be correct.

`\anspic`
`\__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:`

We check that the command is active in the keyanspic environment only if the save-ans key is present, otherwise we return an error. The three arguments are handled by the function `\__enumext_anspic_args:nnn` and stored in the sequence `\l__enumext_anspic_args_seq` which is processed by the keyanspic environment.

```
4491    \NewDocumentCommand \anspic { s o +m }
4492        {
4493            \bool_if:NF \l__enumext_store_active_bool
4494                {
4495                    \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
4496                }
4497            \int_compare:nNnT { \l__enumext_level_int } > { 1 }
4498                {
4499                    \msg_error:nn { enumext } { keyanspic-wrong-level }
4500                }
4501            \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
4502                {
4503                    \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
4504                }
4505            \seq_put_right:Nn \l__enumext_anspic_args_seq
4506                {
4507                    \__enumext_anspic_args:nnn { #1 } { #2 } { #3 }
4508                }
4509        }
```

The `\__enumext_anspic_body_dim:n` function will set the value of `\l__enumext_anspic_body_htdp_dim` equal to the "height plus depth" of the *mandatory argument* if the key label-pos is set "below".

```
4510    \cs_new_protected:Npn \__enumext_anspic_body_dim:n #1
4511        {
4512            \bool_if:NF \l__enumext_anspic_label_above_bool
4513                {
4514                    \IfDocumentMetadataT
4515                        {
4516                            \tag_suspend:n {keyanspic}
4517                        }
4518                    \vbox_set:Nn \l__enumext_anspic_body_box { #1 }
4519                    \dim_set:Nn \l__enumext_anspic_body_htdp_dim
```

```
4520                    {
4521                        \box_ht_plus_dp:N \l__enumext_anspic_body_box
4522                    }
4523                \IfDocumentMetadataT
4524                    {
4525                        \tag_resume:n {keyanspic}
4526                    }
4527            }
4528        }
```

The `\__enumext_anspic_label:nn` function will process inside `\makebox` the *starred argument* '*' and *optional argument* passed to the command. Here we will store the ⟨*label*⟩ and *optional argument* in *prop list* and *sequence* and execute the show-ans, show-pos, font, wrap-label, wrap-ans* and wrap-opt keys.

```
4529  \cs_new_protected:Npn \__enumext_anspic_label:nn #1 #2
4530    {
4531      \makebox[ \l__enumext_anspic_mini_width_dim ][ c ]
4532        {
4533          \bool_if:nTF { #1 }
4534            {
4535              \bool_set_true:N \l__enumext_item_wrap_key_bool
4536              \bool_set_true:N \l__enumext_wrap_label_v_bool
4537              \__enumext_keyans_save_item_opt:n { #2 }
4538              \__enumext_keyans_addto_prop:n { #2 }
4539              \__enumext_keyans_store_ref:
4540              \__enumext_keyans_addto_seq:n { #2 }
4541              \int_gincr:N \g__enumext_check_starred_cmd_int
4542              \__enumext_keyans_show_ans:
4543              \__enumext_keyans_show_pos:
4544              \makebox[ \l__enumext_labelwidth_v_dim ][c]
4545                {
4546                  \tl_use:N \l__enumext_label_font_style_v_tl
4547                  \__enumext_keyans_wrapper_label:n { \l__enumext_label_vi_tl }
4548                }
4549              \skip_horizontal:n { \l__enumext_labelsep_v_dim }
4550              \__enumext_keyans_show_item_opt:
4551            }
4552            {
4553              \bool_set_false:N \l__enumext_item_wrap_key_bool
4554              \tl_use:N \l__enumext_label_font_style_v_tl
4555              \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl }
4556            }
4557        }
4558    }
```

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.

```
4559  \cs_new_protected:Npn \__enumext_anspic_label_pos:nnn #1 #2 #3
4560    {
4561      \stepcounter { enumXvi }
4562      \__enumext_anspic_body_dim:n { #3 }
4563      \bool_if:NTF \l__enumext_anspic_label_above_bool
4564        {
4565          \__enumext_anspic_label:nn { #1 } { #2 }
4566        }
4567        {
4568          \raisebox
4569            {
4570              -\dim_eval:n
4571                {
4572                  \l__enumext_anspic_label_htdp_dim
4573                  + \l__enumext_anspic_body_htdp_dim
4574                  + \box_dp:N \strutbox
4575                  + \l__enumext_anspic_label_sep_skip
4576                }
4577            }
4578            [ 0pt ] [ 0pt ]
4579            {
4580              \__enumext_anspic_label:nn { #1 } { #2 }
4581            }
4582        }
4583    }
4584  %
```

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.

```
4585 \cs_new_protected:Nn \__enumext_anspic_args:nnn
4586   {
4587     \__enumext_anspic_start_list_tag:
4588     \__enumext_anspic_label_pos:nnn { #1 } { #2 } { #3 }
4589     \__enumext_anspic_stop_start_list_tag:
4590     \bool_if:NTF \l__enumext_anspic_label_above_bool
4591       {
4592         \\[\l__enumext_anspic_label_sep_skip] #3
4593       }
4594       {
4595         \\ #3
4596       }
4597     \__enumext_anspic_stop_list_tag:
4598   }
```

The value {⟨*n° upper, n° lower*⟩} 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.

```
4599 \cs_new_protected:Nn \__enumext_anspic_print:n
4600   {
4601     \clist_map_function:nN { #1 } \__enumext_anspic_row:n
4602   }
4603 \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.

```
4604 \cs_new_protected:Nn \__enumext_anspic_row:n
4605   {
4606     \dim_set:Nn \l__enumext_anspic_mini_width_dim { \linewidth / #1 }
4607     \int_set:Nn \l__enumext_anspic_above_int { \l__enumext_anspic_below_int }
4608     \int_set:Nn \l__enumext_anspic_below_int { \l__enumext_anspic_above_int + #1 }
4609     \int_step_inline:nnn
4610       { \l__enumext_anspic_above_int + 1 }
4611       { \l__enumext_anspic_below_int }
4612       {
4613         \IfDocumentMetadataT
4614           {
4615             \tag_suspend:n {minipage}
4616           }
4617         \begin{minipage}[ \l__enumext_anspic_mini_pos_str ]{ \l__enumext_anspic_mini_width_dim }
4618           \centering
4619           \seq_item:Nn \l__enumext_anspic_args_seq { ##1 }
4620         \end{minipage}
4621         \IfDocumentMetadataT
4622           {
4623             \tag_resume:n {minipage}
4624           }
4625       }
4626     \par
4627   }
```

The __enumext_anspic_exec: function will execute all the code in the \anspic command in the second argument of the keyanspic environment definition. If the key layout-sty is not set, everything will be printed on a *single line*.

```
4628 \cs_new_protected:Nn \__enumext_anspic_exec:
4629   {
4630     \tl_if_empty:NTF \l__enumext_anspic_layout_style_tl
4631       {
4632         \__enumext_anspic_print:e { \seq_count:N \l__enumext_anspic_args_seq }
4633       }
4634       {
4635         \__enumext_anspic_print:V \l__enumext_anspic_layout_style_tl
4636       }
4637   }
```

(*End of definition for* \anspic *and others. This function is documented on page 18.*)

13.46 The horizontal environments

Generating *horizontal list environments* is NOT as simple as standard LATEX list environments. The fundamental part of the code is adapted from the `shortlst` package to a more modern version using `expl3`. It is not possible to redefine `\item` and `\makelabel` using `\RenewDocumentCommand` as in the vertical *non starred* versions.

To achieve the *horizontal list environments* we will capture the `\item` command and the ⟨*content*⟩ of this in *horizontal box* using `\makebox` for the `label` and a `minipage` environment for the ⟨*content*⟩ passed to `\item`, we will also add the *optional argument* (⟨*number*⟩) to `\item` to be able to *join columns* horizontally, in simple terms, we want `\item` to behave in the same way as in the `enumext` environment but adding an *first optional argument* (⟨*number*⟩).

A side effect is the limitation of using `\item` in this way *without* using `\RenewDocumentCommand`, which loses the original definition and affects the *standard list environments* provided by LATEX and any environment defined using base `list` environment, including: `itemize`, `enumerate`, `description`, `quote`, `quotation`, `verse`, `center`, `flushleft`, `flushright`, `verbatim`, `tabbing`, `trivlist`, `list` and all environments created with `\newtheorem`.

💣 One way to get around this is to use something like:

```
\AddToHook{env/enumerate/before}{recover original \item definition}
```

inside `minipage`, but in my partial tests this does not have the desired effect and the vertical and horizontal spacing is distorted. For now this will remain as a limitation and I will see if it is feasible to implement it in the future.

🏷 For compatibility with the *tagged* PDF we close the environments according to the presence or not of the `mini-env` key.

13.46.1 Functions for item box width

`\__enumext_starred_columns_set_vii:`
`\__enumext_starred_columns_set_viii:`

We set the default value for the *width of the box* containing the ⟨*content*⟩ of the items for `enumext*` environment.

```
4638 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
4639   {
4640     \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
4641       {
4642         \dim_set:Nn \l__enumext_columns_sep_vii_dim
4643           {
4644             ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
4645             / \l__enumext_columns_vii_int
4646           }
4647       }
4648     \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - 1 }
4649     \dim_set:Nn \l__enumext_item_width_vii_dim
4650       {
4651         ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
4652         / \l__enumext_columns_vii_int
4653         - \l__enumext_labelwidth_vii_dim
4654         - \l__enumext_labelsep_vii_dim
4655       }
```

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

```
4656         \dim_compare:nNnT { \l__enumext_rightmargin_vii_dim } > { \c_zero_dim }
4657           {
4658             \dim_sub:Nn \l__enumext_item_width_vii_dim
4659               {
4660                 ( \l__enumext_rightmargin_vii_dim * \l__enumext_tmpa_vii_int )
4661                 / \l__enumext_columns_vii_int
4662               }
4663             \dim_add:Nn \l__enumext_columns_sep_vii_dim
4664               {
4665                 \l__enumext_rightmargin_vii_dim
4666               }
4667           }
4668   }
```

Same implementation for the `keyans*` environment.

```
4669 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
4670   {
4671     \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
4672       {
4673         \dim_set:Nn \l__enumext_columns_sep_viii_dim
4674           {
4675             ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
4676             / \l__enumext_columns_viii_int
4677           }
4678       }
```

```
4679        \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - 1 }
4680        \dim_set:Nn \l__enumext_item_width_viii_dim
4681          {
4682            ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
4683            / \l__enumext_columns_viii_int
4684            - \l__enumext_labelwidth_viii_dim
4685            - \l__enumext_labelsep_viii_dim
4686          }
4687        \dim_compare:nNnT { \l__enumext_rightmargin_viii_dim } > { \c_zero_dim }
4688          {
4689            \dim_sub:Nn \l__enumext_item_width_viii_dim
4690              {
4691                ( \l__enumext_rightmargin_viii_dim * \l__enumext_tmpa_vii_int )
4692                / \l__enumext_columns_viii_int
4693              }
4694            \dim_add:Nn \l__enumext_columns_sep_viii_dim
4695              {
4696                \l__enumext_rightmargin_viii_dim
4697              }
4698          }
4699      }
```

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

13.46.2 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.

```
4700  \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
4701    {
4702      \int_set:Nn \l__enumext_joined_item_vii_int {#1}
4703      \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
4704        {
4705          \msg_warning:nnee { enumext } { item-joined }
4706            { \int_use:N \l__enumext_joined_item_vii_int }
4707            { \int_use:N \l__enumext_columns_vii_int }
4708          \int_set:Nn \l__enumext_joined_item_vii_int
4709            {
4710              \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4711            }
4712        }
4713      \int_compare:nNnT
4714        { \l__enumext_joined_item_vii_int }
4715        >
4716        { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4717        {
4718          \msg_warning:nnee { enumext } { item-joined-columns }
4719            { \int_use:N \l__enumext_joined_item_vii_int }
4720            {
4721              \int_eval:n
4722                { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4723            }
4724          \int_set:Nn \l__enumext_joined_item_vii_int
4725            {
4726              \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4727            }
4728        }
4729      \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { 1 }
4730        {
4731          \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
4732          \int_decr:N \l__enumext_joined_item_aux_vii_int
4733          \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
4734          \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
4735          \dim_set:Nn \l__enumext_joined_width_vii_dim
4736            {
4737              \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
4738              + (  \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
4739                  + \l__enumext_columns_sep_vii_dim
4740                )*\l__enumext_joined_item_aux_vii_int
4741            }
4742          \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
```

```
4743        }
4744        {
4745            \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
4746            \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
4747        }
4748    }
```

Same implementation for the keyans* environment.

```
4749 \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
4750    {
4751        \int_set:Nn \l__enumext_joined_item_viii_int {#1}
4752        \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
4753            {
4754                \msg_warning:nnee { enumext } { item-joined }
4755                    { \int_use:N \l__enumext_joined_item_viii_int }
4756                    { \int_use:N \l__enumext_columns_viii_int }
4757                \int_set:Nn \l__enumext_joined_item_viii_int
4758                    {
4759                        \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4760                    }
4761            }
4762        \int_compare:nNnT
4763            { \l__enumext_joined_item_viii_int }
4764                >
4765            { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4766            {
4767                \msg_warning:nnee { enumext } { item-joined-columns }
4768                    { \int_use:N \l__enumext_joined_item_viii_int }
4769                    {
4770                        \int_eval:n
4771                            { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4772                    }
4773                \int_set:Nn \l__enumext_joined_item_viii_int
4774                    {
4775                        \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4776                    }
4777            }
4778        \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { 1 }
4779            {
4780                \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
4781                \int_decr:N \l__enumext_joined_item_aux_viii_int
4782                \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
4783                \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
4784                \dim_set:Nn \l__enumext_joined_width_viii_dim
4785                    {
4786                        \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
4787                        + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
4788                            + \l__enumext_columns_sep_viii_dim
4789                        )*\l__enumext_joined_item_aux_viii_int
4790                    }
4791                \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
4792            }
4793            {
4794                \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
4795                \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
4796            }
4797    }
```

(*End of definition for* __enumext_starred_joined_item_vii:n *and* __enumext_starred_joined_item_viii:n.)

13.46.3   Functions for mini-env, mini-right and mini-right* keys

__enumext_start_mini_vii:
__enumext_stop_mini_vii:

The implementation of the mini-env key support is almost identical to the one used in the enumext and keyans environments, the difference is that the __enumext_mini_page environment on the *"right side"* is executed *"after"* closing the environment, so it is necessary to make a global copy of the variable \l__-enumext_minipage_right_vii_dim in the variable \g__enumext_minipage_right_vii_dim.

```
4798 \cs_new_protected:Nn \__enumext_start_mini_vii:
4799    {
4800        \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
4801            {
4802                \dim_set:Nn \l__enumext_minipage_left_vii_dim
4803                    {
```

```
4804              \linewidth
4805              - \l__enumext_minipage_right_vii_dim
4806              - \l__enumext_minipage_hsep_vii_dim
4807            }
4808          \bool_set_true:N \l__enumext_minipage_active_vii_bool
4809          \dim_gset_eq:NN
4810            \g__enumext_minipage_right_vii_dim
4811            \l__enumext_minipage_right_vii_dim
4812          \__enumext_mini_addvspace_vii:
4813          \nointerlineskip\noindent
4814          \__enumext_mini_page{ \l__enumext_minipage_left_vii_dim }
4815        }
4816      }
```

The function __enumext_stop_mini_vii: closes the __enumext_mini_page environment on the *"left side"*, applies \hfill and set the variable \g__enumext_minipage_active_vii_bool to *"true"* which will be used in the function __enumext_after_env:nn to execute the minipage on the *"right side"*. At this point we will execute the __enumext_stop_list: and __enumext_stop_store_level_vii: functions stopping the list environment and the level saving mechanism for storage in *sequence* of the \anskey command and anskey* environment. This function is passed to the __enumext_after_list_vii: function in the second part of the enumext* environment definition (§13.47).

```
4817 \cs_new_protected:Nn \__enumext_stop_mini_vii:
4818   {
4819     \bool_if:NTF \l__enumext_minipage_active_vii_bool
4820       {
4821         \__enumext_stop_list:
4822         \__enumext_stop_store_level_vii:
4823         \IfDocumentMetadataT { \tag_resume:n {enumext*} }
4824         \end__enumext_mini_page
4825         \hfill
4826         \bool_gset_true:N \g__enumext_minipage_active_vii_bool
4827       }
4828       {
4829         \__enumext_stop_list:
4830         \__enumext_stop_store_level_vii:
4831       }
4832   }
```

(*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.

```
4833 \__enumext_after_env:nn {enumext*}
4834   {
4835     \bool_if:NT \g__enumext_minipage_active_vii_bool
4836       {
4837         \__enumext_minipage:w [ t ] { \g__enumext_minipage_right_vii_dim }
4838           \legacy_if_gset_false:n { @minipage }
4839           \skip_vertical:N \c_zero_skip
4840           \par\addvspace { \g__enumext_minipage_right_skip }
4841           \bool_if:NF \g__enumext_minipage_center_vii_bool
4842             {
4843               \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
4844                 {
4845                   \centering
4846                 }
4847             }
4848           \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
4849             {
4850               \tl_use:N \g__enumext_miniright_code_vii_tl
4851             }
4852           \box_use_drop:N \l__enumext_miniright_code_vii_box
4853           \skip_vertical:N \c_zero_skip
4854           \__enumext_endminipage:
4855           \par\addvspace{ \g__enumext_minipage_after_skip }
4856       }
4857     \bool_gset_false:N \g__enumext_minipage_active_vii_bool
4858     \bool_gset_true:N \g__enumext_minipage_center_vii_bool
4859     \tl_gclear:N \g__enumext_miniright_code_vii_tl
4860     \dim_gzero:N \g__enumext_minipage_right_vii_dim
```

```
4861        \bool_gset_false:N \g__enumext_starred_bool
4862    }
```

`\__enumext_start_mini_viii:`
`\__enumext_stop_mini_viii:`

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

```
4863 \cs_new_protected:Nn \__enumext_start_mini_viii:
4864    {
4865      \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
4866        {
4867          \dim_set:Nn \l__enumext_minipage_left_viii_dim
4868            {
4869              \linewidth
4870              - \l__enumext_minipage_right_viii_dim
4871              - \l__enumext_minipage_hsep_viii_dim
4872            }
4873          \bool_set_true:N \l__enumext_minipage_active_viii_bool
4874          \dim_gset_eq:NN
4875            \g__enumext_minipage_right_viii_dim
4876            \l__enumext_minipage_right_viii_dim
4877          \__enumext_mini_addvspace_viii:
4878          \nointerlineskip\noindent
4879          \__enumext_mini_page{ \l__enumext_minipage_left_viii_dim }
4880        }
4881    }
4882 \cs_new_protected:Nn \__enumext_stop_mini_viii:
4883    {
4884      \bool_if:NTF \l__enumext_minipage_active_viii_bool
4885        {
4886          \__enumext_stop_list:
4887          \IfDocumentMetadataT { \tag_resume:n {keyans*} }
4888          \end__enumext_mini_page
4889          \hfill
4890          \bool_gset_true:N \g__enumext_minipage_active_viii_bool
4891        }
4892        {
4893          \__enumext_stop_list:
4894        }
4895    }
4896 \__enumext_after_env:nn {keyans*}
4897    {
4898      \bool_if:NT \g__enumext_minipage_active_viii_bool
4899        {
4900          \__enumext_mini_page{ \g__enumext_minipage_right_viii_dim }
4901          \par\addvspace { \g__enumext_minipage_right_skip }
4902          \bool_if:NF \g__enumext_minipage_center_viii_bool
4903            {
4904              \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
4905                {
4906                  \centering
4907                }
4908            }
4909          \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
4910            {
4911              \tl_use:N \g__enumext_miniright_code_viii_tl
4912            }
4913          \box_use_drop:N \l__enumext_miniright_code_viii_box
4914          \end__enumext_mini_page
4915          \par\addvspace{ \g__enumext_minipage_after_skip }
4916        }
4917      \bool_gset_false:N \g__enumext_minipage_active_viii_bool
4918      \bool_gset_true:N \g__enumext_minipage_center_viii_bool
4919      \tl_gclear:N \g__enumext_miniright_code_viii_tl
4920      \dim_gzero:N \g__enumext_minipage_right_viii_dim
4921    }
```

(*End of definition for* `\__enumext_start_mini_viii:` *and* `\__enumext_stop_mini_viii:`.)

13.47   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.

```
4922 \NewDocumentEnvironment{enumext*}{ o }
4923   {
4924     \__enumext_safe_exec_vii:
4925     \__enumext_parse_keys_vii:n {#1}
4926     \__enumext_before_list_vii:
4927     \__enumext_start_store_level_vii:
4928     \__enumext_start_list:nn { }
4929       {
4930         \__enumext_list_arg_two_vii:
4931         \__enumext_before_keys_exec_vii:
4932       }
4933     \IfDocumentMetadataT { \tag_suspend:n {enumext*} }
4934     \__enumext_starred_columns_set_vii:
4935     \item[] \scan_stop:
4936     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_first_item_tmp_vii:
4937     \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
4938     \ignorespaces
4939   }
4940   {
4941     \IfDocumentMetadataT { \tag_struct_end:n {tag=text-unit} }
4942     \__enumext_stop_item_tmp_vii:
4943     \__enumext_remove_extra_parsep_vii:
4944     \__enumext_after_list_vii:
4945   }
```

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

__enumext_safe_exec_vii: We will first call the function __enumext_is_not_nested: which sets \g__enumext_starred_bool to true if we are NOT nested within enumext, then call the function __enumext_internal_mini_page: to create the environment __enumext_mini_page, we will increment \l__enumext_level_h_int to restrict nesting of the environment, set \l__enumext_starred_bool to true and finally call the function __enumext_is_on_first_level: which sets \l__enumext_starred_first_bool to true if we are not nested, allowing the *"storage system"* to be used.

```
4946 \cs_new_protected:Nn \__enumext_safe_exec_vii:
4947   {
4948     \__enumext_is_not_nested:
4949     \__enumext_internal_mini_page:
4950     \int_incr:N \l__enumext_level_h_int
4951     \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
4952       {
4953         \msg_error:nn { enumext } { nested }
4954       }
4955     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
4956       {
4957         \msg_error:nnn { enumext } { nested-horizontal } { keyans* }
4958       }
4959     \bool_set_true:N \l__enumext_starred_bool
4960     \bool_set_false:N \l__enumext_standar_bool
4961     \__enumext_is_on_first_level:
4962   }
```

(*End of definition for* __enumext_safe_exec_vii:.)

__enumext_parse_keys_vii:n   We will first check the state of the variable \l__enumext_resume_count_vii_bool set by the key resume and call the function __enumext_resume_last_counter: if it is "true", then we will clear the variable \l__enumext_series_name_str used by the key series, process the environment [⟨*key = val*⟩] and execute the function __enumext_parse_series:n 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.

🪀 Here it is necessary to check the status of \l__enumext_resume_count_vii_bool in case the key resume is set using \setenumext{enumext*}{resume}.

```
4963 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
4964   {
4965     \bool_if:NT \l__enumext_resume_count_vii_bool
4966       {
4967         \__enumext_resume_last_counter:
```

```
4968          }
4969       \tl_if_novalue:nF {#1}
4970         {
4971            \str_clear:N \l__enumext_series_name_str
4972            \keys_set:nn { enumext / enumext* } {#1}
4973            \bool_if:NF \l__enumext_print_keyans_cmd_bool
4974              {
4975                 \__enumext_parse_series:n {#1}
4976              }
4977            \__enumext_store_active_keys_vii:n {#1}
4978         }
4979       }
```

(*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*.

```
4980  \cs_new_protected:Nn \__enumext_before_list_vii:
4981    {
4982       \__enumext_vspace_above_vii:
4983       \__enumext_check_ans_active:
4984       \__enumext_before_args_exec_vii:
4985       \__enumext_start_mini_vii:
4986    }
```

(*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: (§13.46.3) used by the keys mini-env, mini-right and mini-right*, then to the functions __enumext_after_stop_-list_vii: used by the key after, __enumext_check_ans_key_hook: used by the key check-ans, __enumext_vspace_below_vii: used by the keys below and below*. Finally set \l__enumext_-starred_bool to false and call the __enumext_resume_save_counter: function used by the series, resume and resume* keys.

```
4987  \cs_new_protected:Nn \__enumext_after_list_vii:
4988    {
4989       \__enumext_stop_mini_vii:
4990       \__enumext_after_stop_list_vii:
4991       \__enumext_check_ans_key_hook:
4992       \__enumext_vspace_below_vii:
4993       \bool_set_false:N \l__enumext_starred_bool
4994       \bool_if:NF \l__enumext_print_keyans_cmd_bool
4995         {
4996            \__enumext_starred_save_counter:
4997         }
4998    }
```

(*End of definition for* __enumext_after_list_vii:.)

__enumext_start_store_level_vii: The __enumext_start_store_level_vii: and __enumext_stop_store_level_vii: functions ac-
__enumext_stop_store_level_vii: tivate the *"storing structure"* mechanism in *sequence* for \anskey command and anskey* environment if enumext* are nested in enumext.

```
4999  \cs_new_protected:Nn \__enumext_start_store_level_vii:
5000    {
5001       \bool_if:NT \l__enumext_store_active_bool
5002         {
5003            \int_compare:nNnT { \l__enumext_level_int } > { 0 }
5004              {
5005                 \__enumext_store_level_open_vii:
5006              }
5007         }
5008    }
5009  \cs_new_protected:Nn \__enumext_stop_store_level_vii:
5010    {
5011       \bool_if:NT \l__enumext_store_active_bool
5012         {
5013            \int_compare:nNnT { \l__enumext_level_int } > { 0 }
5014              {
```

```
5015          \__enumext_store_level_close_vii:
5016        }
5017      }
5018    }
```

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

13.47.1   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.

```
5019 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_vii:
5020   {
5021     \skip_horizontal:n
5022       {
5023         -\l__enumext_labelwidth_vii_dim - \l__enumext_labelsep_vii_dim
5024       }
5025     \ignorespaces
5026   }
```

(*End of definition for* __enumext_first_item_tmp_vii:.)

__enumext_start_item_tmp_vii:
__enumext_item_peek_args_vii:
__enumext_joined_item_vii:w
__enumext_standar_item_vii:w
__enumext_starred_item_vii:w
__enumext_starred_item_vii_aux_i:w
__enumext_starred_item_vii_aux_ii:w
__enumext_starred_item_vii_aux_iii:w

First we will call the function __enumext_stop_item_tmp_vii: that we will redefine later, we will increment the value of \l__enumext_item_column_pos_vii_int that will count the item's by rows and the value of \g__enumext_item_count_all_vii_int that will count the total of item's in the environment. After that we will call the function __enumext_item_peek_args_vii: that will handle the arguments passed to \item.

```
5027 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
5028   {
5029     \__enumext_stop_item_tmp_vii:
5030     \int_incr:N \l__enumext_item_column_pos_vii_int
5031     \int_gincr:N \g__enumext_item_count_all_vii_int
5032     \__enumext_item_peek_args_vii:
5033   }
```

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).

```
5034 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
5035   {
5036     \peek_meaning:NTF (
5037       { \__enumext_joined_item_vii:w }
5038       { \__enumext_joined_item_vii:w (1) }
5039   }
```

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.

```
5040 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
5041   {
5042     \__enumext_starred_joined_item_vii:n {#1}
5043     \peek_meaning_remove:NTF *
5044       { \__enumext_starred_item_vii:w  }
5045       { \__enumext_standar_item_vii:w }
5046   }
```

The function __enumext_standar_item_vii:w will first look for the argument "[", if present it will set the state of the variable \l__enumext_wrap_label_opt_vii_bool equal to the state of the variable \l__enumext_wrap_label_opt_vii_bool handled by the key wrap-label* and finally execute the *non-enumerated* version \item[⟨*custom*⟩] by means of the function __enumext_start_item_vii:w, otherwise we will set the value of the variable \l__enumext_wrap_label_vii_bool handled by the wrap-label key to true and set the switch \if@noitemarg to true to execute the enumerated version of \item by means of the function __enumext_start_item_vii:w [\l__enumext_label_vii_tl].

```
5047 \cs_new_protected:Npn \__enumext_standar_item_vii:w
5048   {
5049     \bool_set_false:N \l__enumext_item_starred_vii_bool
5050     \peek_meaning:NTF [
5051       {
5052         \bool_set_eq:NN \l__enumext_wrap_label_vii_bool \l__enumext_wrap_label_opt_vii_bool
5053         \__enumext_start_item_vii:w
```

```
5054                }
5055              {
5056                \bool_set_true:N \l__enumext_wrap_label_vii_bool
5057                \legacy_if_set_true:n { @noitemarg }
5058                \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ] \ignorespaces
5059              }
5060          }
```

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⟩]`.

```
5061   \cs_new_protected:Npn \__enumext_starred_item_vii:w
5062     {
5063       \bool_set_true:N \l__enumext_item_starred_vii_bool
5064       \bool_set_true:N \l__enumext_wrap_label_vii_bool
5065       \peek_meaning:NTF [
5066         { \__enumext_starred_item_vii_aux_i:w }
5067         { \__enumext_starred_item_vii_aux_ii:w }
5068     }
5069   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
5070     {
5071       \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
5072       \__enumext_starred_item_vii_aux_ii:w
5073     }
5074   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
5075     {
5076       \peek_meaning:NTF [
5077         { \__enumext_starred_item_vii_aux_iii:w }
5078         {
5079           \dim_set_eq:NN \l__enumext_item_symbol_sep_vii_dim \l__enumext_labelsep_vii_dim
5080           \legacy_if_set_true:n { @noitemarg }
5081           \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ] \ignorespaces
5082         }
5083     }
5084   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
5085     {
5086       \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
5087       \legacy_if_set_true:n { @noitemarg }
5088       \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ] \ignorespaces
5089     }
```

(*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`

```
5090   \cs_new_protected_nopar:Npn \__enumext_fake_make_label_vii:n #1
5091     {
5092       \legacy_if:nT { @noitemarg }
5093         {
5094           \legacy_if_set_false:n { @noitemarg }
5095           \legacy_if:nT { @nmbrlist }
5096             {
5097               \IfDocumentMetadataT
5098                 {
5099                   \bool_if:NT \l__enumext_hyperref_bool
5100                     {
5101                       \legacy_if_set_true:n { @hyper@item }
5102                     }
5103                 }
5104               \refstepcounter{enumXvii}
5105               \bool_if:NT \l__enumext_check_answers_bool
5106                 {
```

```
5107                     \int_gincr:N \g__enumext_item_number_int
5108                     \bool_set_true:N \l__enumext_item_number_bool
5109                   }
5110               }
5111           }
5112       \bool_if:NT \l__enumext_item_starred_vii_bool
5113         {
5114           \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
5115             {
5116               \tl_gset_eq:NN
5117                 \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
5118             }
5119           \mode_leave_vertical:
5120           \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
5121           \hbox_overlap_left:n { \g__enumext_item_symbol_aux_vii_tl }
5122           \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
5123           \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
5124         }
5125       \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
5126         {
5127           \tl_use:N \l__enumext_label_font_style_vii_tl
5128           \bool_if:NTF \l__enumext_wrap_label_vii_bool
5129             {
5130               \__enumext_wrapper_label_vii:n {#1}
5131             }
5132             { #1 }
5133         }
5134       \skip_horizontal:N \l__enumext_labelsep_vii_dim \ignorespaces
5135     }
```

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

13.47.2 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
__enumext_stop_item_vii:

The first thing we will do is set the value of __enumext_stop_item_tmp_vii: equal to __enumext_-stop_item_vii: which we will define later, after that we will start capturing \item and *"item content"* in a *horizontal box* where the width will be \itemwidth plus \labelwidth plus \labelsep.

```
5136 \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
5137   {
5138     \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
5139     \hbox_set_to_wd:Nnw \l__enumext_item_text_vii_box
5140       {
5141         \l__enumext_joined_width_vii_dim
5142         + \l__enumext_labelwidth_vii_dim
5143         + \l__enumext_labelsep_vii_dim
5144       }
```

Redefine the \footnote command.

```
5145         \__enumext_renew_footnote_starred:
```

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

```
5146         \__enumext_start_list_tag:n {enumext*}
5147         \__enumext_fake_make_label_vii:n {#1}
5148         \__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.

```
5149         \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
5150           \dim_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
5151           \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
5152           \__enumext_unskip_unkern:
5153           \__enumext_unskip_unkern:
5154           \skip_horizontal:n { -\l__enumext_listparindent_vii_dim } \ignorespaces
5155           \tl_use:N \l__enumext_fake_item_indent_vii_tl
5156           \tl_use:N \l__enumext_after_list_args_vii_tl
5157       }
```

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*.

```
5158 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
5159   {
5160       \__enumext_endminipage:
5161     \__enumext_stop_list_tag:n {enumext*}
5162     \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*,.

```
5163       \int_set:Nn \hbadness { 10000 }
5164       \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`.

```
5165       \int_compare:nNnTF
5166         { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
5167         {
5168           \par\noindent
5169           \int_zero:N \l__enumext_item_column_pos_vii_int
5170         }
5171         {
5172           \skip_horizontal:N \l__enumext_columns_sep_vii_dim
5173         }
5174   }
```

(*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*⟩.

```
5175 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
5176   {
5177     \int_compare:nNnT
5178       {
5179         \int_mod:nn
5180           {  \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
5181       }
5182       =
5183       { 0 }
5184       {
5185         \para_end:
5186         \skip_vertical:n { -\l__enumext_itemsep_vii_skip }
5187         \skip_vertical:N \c_zero_skip
5188         \int_gzero:N \g__enumext_item_count_all_vii_int
5189       }
5190   }
```

(*End of definition for* `\__enumext_remove_extra_parsep_vii:`.)

As we don't want our check to be executed `check-ans` by levels but on the complete list, we will take it out of the `enumext*` environment using the *"hook"* function `\__enumext_after_env:nn`.

```
5191 \__enumext_after_env:nn {enumext*}
5192   {
5193     \__enumext_execute_after_env:
5194   }
```

13.48   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.

```
5195 \NewDocumentEnvironment{keyans*}{ o }
5196   {
5197     \__enumext_safe_exec_viii:
5198     \__enumext_parse_keys_viii:n {#1}
5199     \__enumext_before_list_viii:
5200     \__enumext_start_list:nn { }
5201       {
5202         \__enumext_list_arg_two_viii:
5203         \__enumext_before_keys_exec_viii:
5204       }
```

```
5205    \IfDocumentMetadataT { \tag_suspend:n {keyans*} }
5206    \__enumext_starred_columns_set_viii:
5207    \item[] \scan_stop:
5208    \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_first_item_tmp_viii:
5209    \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
5210    \ignorespaces
5211    }
5212    {
5213    \IfDocumentMetadataT { \tag_struct_end:n {tag=text-unit} }
5214    \__enumext_stop_item_tmp_viii:
5215    \__enumext_remove_extra_parsep_viii:
5216    \__enumext_check_starred_cmd:n { item }
5217    \__enumext_after_list_viii:
5218    }
```

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

__enumext_safe_exec_viii: The __enumext_safe_exec_viii: function will first check if the save-ans key is active and only when this is true the environment will be available, it will increment the value of \l__enumext_keyans_level_h_int and return an error message when we are nesting the environment, then it will call the __enumext_-keyans_name_and_start: function in charge of saving the name of the environment and the line it is running on, then it will check if we are trying to nest keyans* in enumext* returning an error and we will set \l__enumext_starred_bool to true, finally we will check if we are within the appropriate level within the enumext environment.

```
5219  \cs_new_protected:Nn \__enumext_safe_exec_viii:
5220    {
5221      \bool_if:NF \l__enumext_store_active_bool
5222        {
5223          \msg_error:nnnn { enumext } { wrong-place }{ keyans* }{ save-ans }
5224        }
5225      \int_incr:N \l__enumext_keyans_level_h_int
5226      \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
5227        {
5228          \msg_error:nn { enumext } { nested }
5229        }
5230      \__enumext_keyans_name_and_start:
5231      \bool_if:NT \l__enumext_starred_bool
5232        {
5233          \msg_error:nnn { enumext } { nested-horizontal } { enumext* }
5234        }
5235      \bool_set_true:N \l__enumext_starred_bool
5236      % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
5237      \bool_set_false:N \l__enumext_store_active_bool
5238      \int_compare:nNnT { \l__enumext_level_int } > { 1 }
5239        {
5240          \msg_error:nn { enumext } { keyans-wrong-level }
5241        }
5242    }
```

(*End of definition for* __enumext_safe_exec_viii:.)

__enumext_parse_keys_viii:n   Parse [⟨*key = val*⟩] for keyans*.

```
5243  \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
5244    {
5245      \tl_if_novalue:nF {#1}
5246        {
5247          \keys_set:nn { enumext / keyans* } {#1}
5248        }
5249    }
```

(*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.

```
5250  \cs_new_protected:Nn \__enumext_before_list_viii:
5251    {
5252      \__enumext_vspace_above_viii:
5253      \__enumext_before_args_exec_viii:
5254      \__enumext_start_mini_viii:
5255    }
```

(*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.

```
5256 \cs_new_protected:Nn \__enumext_after_list_viii:
5257   {
5258     \__enumext_stop_mini_viii:
5259     \__enumext_after_stop_list_viii:
5260     \__enumext_vspace_below_viii:
5261   }
```

(*End of definition for* `\__enumext_after_list_viii:`.)

13.48.1 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.

```
5262 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_viii:
5263   {
5264     \skip_horizontal:n
5265       {
5266         -\l__enumext_labelwidth_viii_dim - \l__enumext_labelsep_viii_dim
5267       }
5268     \ignorespaces
5269   }
```

(*End of definition for* `\__enumext_first_item_tmp_viii:`.)

`\__enumext_start_item_tmp_viii:`
`\__enumext_item_peek_args_viii:`
`\__enumext_joined_item_viii:w`
`\__enumext_standar_item_viii:w`

First we will call the function `\__enumext_stop_item_tmp_viii:` that we will redefine later, we will increment the value of `\l__enumext_item_column_pos_viii_int` that will count the item's by rows and the value of `\g__enumext_item_count_all_viii_int` that will count the total of item's in the environment. After that we will call the function `\__enumext_item_peek_args_viii:` that will handle the arguments passed to `\item`.

```
5270 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
5271   {
5272     \__enumext_stop_item_tmp_viii:
5273     \int_incr:N \l__enumext_item_column_pos_viii_int
5274     \int_gincr:N \g__enumext_item_count_all_viii_int
5275     \__enumext_item_peek_args_viii:
5276   }
```

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).

```
5277 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
5278   {
5279     \peek_meaning:NTF (
5280       { \__enumext_joined_item_viii:w }
5281       { \__enumext_joined_item_viii:w (1) }
5282   }
```

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`.

```
5283 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
5284   {
5285     \__enumext_starred_joined_item_viii:n {#1}
5286     \peek_meaning_remove:NTF *
5287       { \__enumext_starred_item_viii:w }
5288       { \__enumext_standar_item_viii:w }
5289   }
```

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 ]`.

```
5290  \cs_new_protected:Npn \__enumext_standar_item_viii:w
5291    {
5292      \bool_set_false:N \l__enumext_item_starred_viii_bool
5293      \bool_set_false:N \l__enumext_item_wrap_key_bool
5294      \peek_meaning:NTF [
5295        {
5296          \bool_set_eq:NN \l__enumext_wrap_label_viii_bool \l__enumext_wrap_label_opt_viii_bool
5297          \__enumext_start_item_viii:w
5298        }
5299        {
5300          \bool_set_true:N \l__enumext_wrap_label_viii_bool
5301          \legacy_if_set_true:n { @noitemarg }
5302          \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ] \ignorespaces
5303        }
5304    }
```

(*End of definition for* `\__enumext_start_item_tmp_viii:` *and others.*)

`\__enumext_starred_item_viii:w`
`\__enumext_starred_item_viii_aux_i:w`
`\__enumext_starred_item_viii_aux_ii:w`
`\__enumext_keyans_starred_item_star:`

The function `\__enumext_starred_item_viii:w` together with the specified auxiliary functions aux_i:w and aux_ii:w execute `\item*` and `\item*[`⟨*content*⟩`]`.

```
5305  \cs_new_protected:Npn \__enumext_starred_item_viii:w
5306    {
5307      \bool_set_true:N \l__enumext_item_starred_viii_bool
5308      \bool_set_true:N \l__enumext_item_wrap_key_bool
5309      \bool_set_true:N \l__enumext_wrap_label_viii_bool
5310      \peek_meaning:NTF [
5311        { \__enumext_starred_item_viii_aux_i:w }
5312        { \__enumext_starred_item_viii_aux_ii:w }
5313    }
```

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`.

```
5314  \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
5315    {
5316      \tl_clear:N \l__enumext_store_current_label_tl
5317      \tl_if_novalue:nF { #1 }
5318        {
5319          \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_viii_tl
5320            {
5321              \tl_put_right:NV \l__enumext_store_current_label_tl \l__enumext_store_keyans_item_opt_
5322              \tl_put_right:Nn \l__enumext_store_current_label_tl { #1 }
5323            }
5324          \tl_set:Nn \l__enumext_store_current_opt_arg_tl { #1 }
5325        }
5326      \__enumext_starred_item_viii_aux_ii:w
5327    }
5328  \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
5329    {
5330      \legacy_if_set_true:n { @noitemarg }
5331      \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ] \ignorespaces
5332    }
```

The function `\__enumext_keyans_starred_item_star:` will be in charge of storing the current ⟨*label*⟩ for `\item*` followed by the [⟨*content*⟩] for `\item*[`⟨*content*⟩`]` if present in the *sequence* and *prop list* set by the save-ans key. In this same function the keys show-ans, show-pos, mark-sep and save-ref are implemented.

```
5333  \cs_new_protected:Nn \__enumext_keyans_starred_item_star:
5334    {
5335      \tl_put_left:Ne \l__enumext_store_current_label_tl { \l__enumext_label_viii_tl }
5336      \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
5337      \__enumext_keyans_store_ref:
5338      \tl_put_left:Nn \l__enumext_store_current_label_tl { \item }
```

```
5339        \__enumext_keyans_addto_seq_link:
5340        \int_gincr:N \g__enumext_check_starred_cmd_int
5341        \dim_compare:nNnT { \l__enumext_mark_sym_sep_viii_dim } = { \c_zero_dim }
5342          {
5343            \dim_set:Nn \l__enumext_mark_sym_sep_viii_dim { \l__enumext_labelsep_viii_dim }
5344          }
5345        \bool_if:NT \l__enumext_show_answer_bool
5346          {
5347            \tl_set_eq:NN \l__enumext_mark_answer_sym_tl \l__enumext_mark_answer_sym_viii_tl
5348            \str_set_eq:NN \l__enumext_mark_position_str \l__enumext_mark_position_viii_str
5349            \__enumext_print_keyans_box:NN
5350              \l__enumext_labelwidth_viii_dim \l__enumext_mark_sym_sep_viii_dim
5351          }
5352        \bool_if:NT \l__enumext_show_position_bool
5353          {
5354            \tl_set:Ne \l__enumext_mark_answer_sym_tl
5355              {
5356                \group_begin:
5357                  \exp_not:N \normalfont
5358                  \exp_not:N \footnotesize [ \int_eval:n
5359                    {
5360                      \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
5361                    }
5362                  ]
5363                \group_end:
5364              }
5365            \str_set_eq:NN \l__enumext_mark_position_str \l__enumext_mark_position_viii_str
5366            \__enumext_print_keyans_box:NN
5367              \l__enumext_labelwidth_viii_dim \l__enumext_mark_sym_sep_viii_dim
5368          }
5369      }
```

(*End of definition for __enumext_starred_item_viii:w and others.*)

__enumext_keyans_wraper_label_viii:n

__enumext_fake_make_label_viii:n

The implementation at this is very similar to that of the enumext* environment.

```
5370  \cs_new_protected:Npn \__enumext_keyans_wraper_label_viii:n #1
5371    {
5372      \bool_lazy_all:nT
5373        {
5374          { \bool_if_p:N \l__enumext_wrap_label_viii_bool          }
5375          { \bool_if_p:N \l__enumext_show_answer_bool              }
5376          { \bool_if_p:N \l__enumext_item_wrap_key_bool            }
5377          { \cs_if_exist_p:N \__enumext_keyans_wrapper_item_viii:n }
5378        }
5379        {
5380          \cs_set_eq:NN
5381            \__enumext_wrapper_label_viii:n \__enumext_keyans_wrapper_item_viii:n
5382        }
5383      \bool_if:NTF \l__enumext_wrap_label_viii_bool
5384        {
5385          \__enumext_wrapper_label_viii:n {#1}
5386        }
5387        { #1 }
5388    }
5389  \cs_new_protected_nopar:Npn \__enumext_fake_make_label_viii:n #1
5390    {
5391      \legacy_if:nT { @noitemarg }
5392        {
5393          \legacy_if_set_false:n { @noitemarg }
5394          \legacy_if:nT { @nmbrlist }
5395            {
5396              \refstepcounter{enumXviii}
5397            }
5398        }
5399      \bool_if:NT \l__enumext_item_starred_viii_bool
5400        {
5401          \__enumext_keyans_starred_item_star:
5402        }
5403      \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
5404        {
5405          \tl_use:N \l__enumext_label_font_style_viii_tl
```

```
5406          \__enumext_keyans_wraper_label_viii:n {#1}
5407        }
5408      \skip_horizontal:N \l__enumext_labelsep_viii_dim \ignorespaces
5409    }
```

(*End of definition for* __enumext_keyans_wraper_label_viii:n *and* __enumext_fake_make_label_viii:n.)

13.48.2   Real definition of \item in keyans*

__enumext_start_item_viii:w
__enumext_stop_item_viii:

The implementation at this is very similar to that of the enumext* environment.

```
5410  \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
5411    {
5412      \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
5413      \hbox_set_to_wd:Nnw \l__enumext_item_text_viii_box
5414        {
5415          \l__enumext_joined_width_viii_dim
5416          + \l__enumext_labelwidth_viii_dim
5417          + \l__enumext_labelsep_viii_dim
5418        }
5419      \__enumext_renew_footnote_starred:
5420      \__enumext_start_list_tag:n {keyans*}
5421      \__enumext_fake_make_label_viii:n {#1}
5422      \__enumext_stop_start_list_tag:
5423      \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim  }
5424        \dim_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
5425        \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
5426        \__enumext_unskip_unkern:
5427        \__enumext_unskip_unkern:
5428        \skip_horizontal:n { -\l__enumext_listparindent_viii_dim } \ignorespaces
5429        \tl_use:N \l__enumext_fake_item_indent_viii_tl
5430        \bool_if:NT \l__enumext_item_starred_viii_bool
5431          {
5432            \__enumext_keyans_show_item_opt_viii:
5433          }
5434        \tl_use:N \l__enumext_after_list_args_viii_tl
5435    }
5436  \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
5437    {
5438      \__enumext_endminipage:
5439      \__enumext_stop_list_tag:n {keyans*}
5440      \hbox_set_end:
5441      \int_set:Nn \hbadness { 10000 }
5442      \box_use_drop:N \l__enumext_item_text_viii_box
5443      \int_compare:nNnTF
5444        { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
5445        {
5446          \par\noindent
5447          \int_zero:N \l__enumext_item_column_pos_viii_int
5448        }
5449        {
5450          \skip_horizontal:N \l__enumext_columns_sep_viii_dim
5451        }
5452    }
```

(*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.

```
5453  \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
5454    {
5455      \int_compare:nNnT
5456        {
5457          \int_mod:nn
5458            { \g__enumext_item_count_all_viii_int }
5459            { \l__enumext_columns_viii_int }
5460        }
5461        =
5462        { 0 }
5463        {
5464          \para_end:
5465          \skip_vertical:n { -\l__enumext_itemsep_viii_skip }
5466          \skip_vertical:N \c_zero_skip
5467          \int_gzero:N \g__enumext_item_count_all_viii_int
```

```
5468            }
5469        }
```

(*End of definition for* __enumext_remove_extra_parsep_viii:.)

13.49   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.

```
5470    \NewDocumentCommand \getkeyans { m }
5471        {
5472            \exp_args:Ne \__enumext_getkeyans_aux:n
5473                { \tl_to_str:e { \text_expand:n {#1} } }
5474        }
```

The internal function __enumext_getkeyans_aux:n is in charge of *splitting* the *mandatory argument* using ":". If ":" is omitted it will return an error.

```
5475    \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
5476        {
5477            \str_if_in:nnTF {#1} { : }
5478                {
5479                    \use:e
5480                        {
5481                            \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
5482                                { {##1} {##2} }
5483                        }
5484                    \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
5485                }
5486                { \msg_error:nnn { enumext } { missing-colon } {#1} }
5487        }
```

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*.

```
5488    \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
5489        {
5490            \prop_if_exist:cTF { g__enumext_#1_prop }
5491                {
5492                    \prop_item:cn { g__enumext_#1_prop }{#2}
5493                }
5494                {
5495                    \msg_error:nnn { enumext } { undefined-storage-anskey } {#1}
5496                }
5497        }
```

(*End of definition for* \getkeyans*,* __enumext_getkeyans_aux:n*, and* __enumext_getkeyans:nn*. This function is documented on page 19.*)

13.50   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*⟩].

```
5498    \keys_define:nn  { enumext / print }
5499        {
5500            print*  .code:n    = \keys_precompile:neN { enumext / enumext* }
5501                                    { \__enumext_filter_save_key:n {#1} }
5502                                    \l__enumext_print_keyans_starred_tl, % starred cmd
5503            print*  .initial:n = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5504                                    rightmargin=0pt, listparindent=0pt, nosep, label=\arabic*.,
5505                                    columns=2, first=\small, font=\small },
5506            print-1 .code:n    = \keys_precompile:neN { enumext / level-1 }
5507                                    { \__enumext_filter_save_key:n {#1} }
5508                                    \l__enumext_print_keyans_i_tl,
5509            print-1 .initial:n = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
```

```
5510                                 rightmargin=0pt, listparindent=0pt, nosep, label=\arabic*.,
5511                                 columns=2, first=\small, font=\small },
5512       print-2 .code:n      = \keys_precompile:neN { enumext / level-2 }
5513                                 { \__enumext_filter_save_key:n {#1} }
5514                                 \l__enumext_print_keyans_ii_tl,
5515       print-2 .initial:n  = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5516                                 rightmargin=0pt, listparindent=0pt,  nosep, label=(\alph*),
5517                                 first=\small, font=\small },
5518       print-3 .code:n      = \keys_precompile:neN { enumext / level-3 }
5519                                 { \__enumext_filter_save_key:n {#1} }
5520                                 \l__enumext_print_keyans_iii_tl,
5521       print-3 .initial:n  = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5522                                 rightmargin=0pt, listparindent=0pt, nosep, label=\roman*.,
5523                                 first=\small, font=\small },
5524       print-4 .code:n      = \keys_precompile:neN { enumext / level-4 }
5525                                 { \__enumext_filter_save_key:n {#1} }
5526                                 \l__enumext_print_keyans_iv_tl,
5527       print-4 .initial:n  = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5528                                 rightmargin=0pt, listparindent=0pt, nosep, label=\Alph*.,
5529                                 first=\small, font=\small },
5530       print-* .code:n      = \keys_precompile:neN { enumext / enumext* }
5531                                 { \__enumext_filter_save_key:n {#1} }
5532                                 \l__enumext_print_keyans_vii_tl, % starred nested
5533       print-* .initial:n  = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5534                                 rightmargin=0pt, listparindent=0pt, nosep, label=\arabic*.,
5535                                 first=\small, font=\small },
5536     }
```

🌻 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.

```
5537  \NewDocumentCommand \printkeyans { s O{} m }
5538    {
5539      \group_begin:
5540        \bool_set_true:N \l__enumext_print_keyans_cmd_bool
5541        \tl_use:N \l__enumext_print_keyans_i_tl
5542        \tl_use:N \l__enumext_print_keyans_ii_tl
5543        \tl_use:N \l__enumext_print_keyans_iii_tl
5544        \tl_use:N \l__enumext_print_keyans_iv_tl
5545        \tl_use:N \l__enumext_print_keyans_vii_tl
5546        \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
5547        \bool_set_false:N \l__enumext_print_keyans_cmd_bool
5548      \group_end:
5549    }
```

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.

```
5550  \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
5551    {
5552      \seq_if_exist:cTF { g__enumext_#3_seq }
5553        {
5554          \seq_if_empty:cF { g__enumext_#3_seq }
5555            {
```

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.

```
5556              \bool_if:nTF {#1}
5557                {
5558                  \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
5559                    {
5560                      \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
5561                    }
5562                    {
```

```
5563                        \tl_use:N \l__enumext_print_keyans_starred_tl
5564                        \bool_set_true:N \l__enumext_base_line_fix_bool
5565                        \bool_set_true:N \l__enumext_print_keyans_star_bool
5566                        \begin{enumext*}[#2]
5567                          \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5568                        \end{enumext*}
5569                        \bool_set_false:N \l__enumext_base_line_fix_bool
5570                        \bool_set_false:N \l__enumext_print_keyans_star_bool
5571                      }
5572                  }
```

Otherwise it will open the environment enumext passing the *optional argument* to the *"first level"* then map the *sequence.*

```
5573                  {
5574                    \begin{enumext}[#2]
5575                      \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5576                    \end{enumext}
5577                  }
5578                }
5579            }
5580            {
5581              \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
5582            }
5583    }
```

(*End of definition for* \printkeyans *and* __enumext_printkeyans:nnn. *This function is documented on page 20.*)

13.51   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* and enumext environments so as not to capture ⟨*keys*⟩ that complicate us.

__enumext_filter_level:n
__enumext_filter_level_key:n
__enumext_filter_level_pair:nn

The function __enumext_filter_level:n will be in charge of filtering the ⟨*keys*⟩ passed to the enumext and enumext* environments.

```
5584  \cs_new:Npn \__enumext_filter_level:n #1
5585    {
5586      \use:e
5587        {
5588          \keyval_parse:NNn
5589            \__enumext_filter_level_key:n
5590            \__enumext_filter_level_pair:nn {#1}
5591        }
5592    }
```

The function __enumext_filter_level_key:n will be responsible for filtering the ⟨*keys*⟩ that are passed *"without value"* by excluding the keys resume*, reset and reset* passed to the enumext and enumext* environments.

```
5593  \cs_new:Npn \__enumext_filter_level_key:n #1
5594    {
5595      \str_case:nnF {#1}
5596        {
5597          { resume* } {} { reset } {} { reset* } {}
5598        }
5599        { , { \exp_not:n {#1} } }
5600    }
```

The function __enumext_filter_level_pair:nn will be responsible for filtering the ⟨*keys*⟩ that are passed *"with value"* by excluding the series, resume and save-ans keys passed to the enumext and enumext* environments.

```
5601  \cs_new:Npn \__enumext_filter_level_pair:nn #1#2
5602    {
5603      \str_case:nnF {#1}
5604        {
5605          { series } {}  { save-ans } {} { resume } {}
5606        }
5607        { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
5608    }
```

(*End of definition for* __enumext_filter_level:n, __enumext_filter_level_key:n, *and* __enumext_filter_level_-pair:nn.)

Now define a *"meta families"* of ⟨*keys*⟩ to access from \setenumext.

```
5609  \keys_define:nn { enumext / meta-families }
5610    {
5611      enumext-1 .code:n = {
5612                          \keys_set:ne { enumext / level-1 }
5613                            {
5614                              \__enumext_filter_level:n {#1}
5615                            }
5616                        },
5617      enumext-2 .code:n = {
5618                          \keys_set:ne { enumext / level-2 }
5619                            {
5620                              \__enumext_filter_level:n {#1}
5621                            }
5622                        },
5623      enumext-3 .code:n = {
5624                          \keys_set:ne { enumext / level-3 }
5625                            {
5626                              \__enumext_filter_level:n {#1}
5627                            }
5628                        },
5629      enumext-4 .code:n = {
5630                          \keys_set:ne { enumext / level-4 }
5631                            {
5632                              \__enumext_filter_level:n {#1}
5633                            }
5634                        },
5635      enumext*  .code:n = {
5636                          \keys_set:ne { enumext / enumext* }
5637                            {
5638                              \__enumext_filter_level:n {#1}
5639                            }
5640                        },
5641      keyans    .code:n = { \keys_set:nn { enumext / keyans  } {#1} },
5642      keyans*   .code:n = { \keys_set:nn { enumext / keyans* } {#1} },
5643      print*    .code:n = { \keys_set:nn { enumext / print   } { print*  = {#1} } },
5644      print-1   .code:n = { \keys_set:nn { enumext / print   } { print-1 = {#1} } },
5645      print-2   .code:n = { \keys_set:nn { enumext / print   } { print-2 = {#1} } },
5646      print-3   .code:n = { \keys_set:nn { enumext / print   } { print-3 = {#1} } },
5647      print-4   .code:n = { \keys_set:nn { enumext / print   } { print-4 = {#1} } },
5648      print-*   .code:n = { \keys_set:nn { enumext / print   } { print-* = {#1} } },
5649      unknown   .code:n = { \msg_error:nn { enumext } { unknown-key-family } },
5650    }
```

We store them in the constant sequence \c__enumext_all_families_seq separated by commas.

```
5651  \seq_const_from_clist:Nn \c__enumext_all_families_seq
5652    {
5653      enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
5654      keyans*, print-1, print-2, print-3, print-4, print-*, print*,
5655    }
```

\setenumext Now we define the user command \setenumext.
__enumext_set_parse:n
__enumext_set_error:nn

```
5656  \NewDocumentCommand \setenumext { O{enumext,1} +m }
5657    {
5658      \seq_clear:N \l__enumext_setkey_tmpa_seq
5659      \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
5660      \int_set:Nn \l__enumext_setkey_tmpa_int
5661        {
5662          \seq_count:N \l__enumext_setkey_tmpb_seq
5663        }
5664      \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
5665        {
5666          \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
5667          \seq_map_function:NN \l__enumext_setkey_tmpb_seq \__enumext_set_parse:n
5668          \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
5669            {
5670              \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
5671            }
5672        }
5673        {
5674          \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
5675        }
```

```
5676    \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
5677      { \seq_map_inline:Nn \c__enumext_all_families_seq }
5678      { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
5679      {
5680        \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
5681      }
5682    }
```

Internal functions used by the \setenumext command.

```
5683  \cs_new_protected:Npn \__enumext_set_parse:n #1
5684    {
5685      \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
5686      \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
5687        { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
5688      \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
5689        {
5690          \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
5691            { \tl_trim_spaces:n {#1} }
5692        }
5693        { \__enumext_set_error:nn {#1} { } }
5694    }
5695  \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
5696    { \msg_error:nnnn { 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.*)

13.52   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 Simplify syntax for command that adds .meta key to existing keys (l3keys).

\setenumextmeta
__enumext_key_set_meta:nnn
__enumext_key_def_meta:nnn
__enumext_key_def_meta:Vnn

First we will create a ⟨*keys*⟩ of type .code:n for "*all levels*" of the enumext environment.

```
5697  \int_step_inline:nn { 4 }
5698    {
5699      \keys_define:nn { enumext }
5700        {
5701          #1 .code:n = \str_set:Nn \l__enumext_meta_path_str { level-#1 }
5702          ,#1 .value_forbidden:n = true
5703        }
5704    }
```

And now we define the ⟨*keys*⟩ for the environments using .code:n for the enumext environment and .meta:n for the enumext* environment.

```
5705  \clist_map_inline:nn { enumext }
5706    {
5707      \keys_define:nn { enumext }
5708        {
5709          #1   .code:n = % ignored for now, might do something useful in the future
5710          ,#1   .value_forbidden:n = true
5711          ,#1* .code:n = \str_set:Nn \l__enumext_meta_path_str { #1* }
5712          ,#1* .value_forbidden:n = true
5713        }
5714    }
5715  \keys_define:nn { enumext }
5716    {
5717      * .meta:n = enumext*
5718      ,* .value_forbidden:n = true
5719    }
```

Now we create the user command taking care that unknown cannot be passed as an argument.

```
5720  \NewDocumentCommand \setenumextmeta { s O{enumext,1} m +m }
5721    {
5722      \str_if_eq:eeTF { \tl_trim_spaces:n {#3} } { unknown }
5723        { \msg_error:nn { enumext } { prohibited-unknown } }
5724        {
5725          \bool_if:nTF {#1}
5726            {
5727              \int_step_inline:nn { 4 }
5728                { \__enumext_key_set_meta:nnn { enumext, ##1 } {#3} {#4} }
5729              \__enumext_key_set_meta:nnn { enumext* } {#3} {#4}
5730            }
```

```
5731                    { \__enumext_key_set_meta:nnn {#2} {#3} {#4} }
5732             }
5733       }
```

The internal functions `\__enumext_key_set_meta:nnn` and `\__enumext_key_def_meta:nnn` will check the *optional argument* and create the *"meta-key"*.

```
5734  \cs_new_protected:Npn \__enumext_key_set_meta:nnn #1
5735    {
5736      \keys_set:nn { enumext } {#1}
5737      \__enumext_key_def_meta:Vnn \l__enumext_meta_path_str
5738    }
5739  \cs_new_protected:Npn \__enumext_key_def_meta:nnn #1#2#3
5740    {
5741      \bool_lazy_or:nnTF
5742        { \keys_if_exist_p:nn { enumext / #1} {#2} }
5743        { \keys_if_exist_p:nn { enumext / enumext* } {#2} }
5744        { \msg_error:nnn { enumext } { already-defined } {#2} }
5745        {
5746          \keys_define:nn { enumext / #1 }
5747            {
5748              #2 .meta:n = {#3},
5749              #2 .value_forbidden:n = true
5750            }
5751        }
5752    }
5753  \cs_generate_variant:Nn \__enumext_key_def_meta:nnn { V }
```

(*End of definition for* `\setenumextmeta`*,* `\__enumext_key_set_meta:nnn`*, and* `\__enumext_key_def_meta:nnn`*. This function is documented on page 6.*)

13.53   The command `\foreachkeyans`

The command `\foreachkeyans` will execute a *loop* over the *prop list* and return its contents. The implementation code is adapted from the answer provided by Enrico Gregorio (@egreg) in Expand a `.cs` defined by key inside the function.

`\foreachkeyans`
`\__enumext_parse_foreach_keys:nn`
`\__enumext_parse_foreach_keys:n`
`\__enumext_foreach_keyans:nn`
`\__enumext_foreach_add_body:n`

We define a set of ⟨*keys*⟩ for command and we will save the default values of these in `\g__enumext_-foreach_default_keys_tl` to avoid the use of group.

```
5754  \keys_define:nn { enumext / foreach }
5755    {
5756      before   .tl_set:N  = \l__enumext_foreach_before_tl,
5757      before   .value_required:n = true,
5758      after    .tl_set:N  = \l__enumext_foreach_after_tl,
5759      after    .value_required:n = true,
5760      start    .int_set:N = \l__enumext_foreach_start_int,
5761      start    .value_required:n = true,
5762      stop     .int_set:N = \l__enumext_foreach_stop_int,
5763      stop     .value_required:n = true,
5764      step     .int_set:N = \l__enumext_foreach_step_int,
5765      step     .value_required:n = true,
5766      wrapper .cs_set_protected:Np = \__enumext_foreach_wrapper:n #1,
5767      wrapper .value_required:n = true,
5768      sep      .tl_set:N  = \l__enumext_foreach_sep_tl,
5769      sep      .value_required:n = true,
5770      unknown .code:n    = { \__enumext_parse_foreach_keys:n {#1} }
5771    }
5772  \keys_precompile:nnN { enumext / foreach }
5773    {
5774      before={},after={},start=1,step=1,stop=0,wrapper=#1,sep={; }
5775    }
5776    \l__enumext_foreach_default_keys_tl
```

Functions for handling unknown ⟨*keys*⟩.

```
5777  \cs_new_protected:Npn \__enumext_parse_foreach_keys:nn #1#2
5778    {
5779      \tl_if_blank:nTF {#2}
5780        {
5781          \msg_error:nnn { enumext } { for-key-unknown } {#1}
5782        }
5783        {
5784          \msg_error:nnnn { enumext } { for-key-value-unknown } {#1} {#2}
5785        }
5786    }
```

```
5787  \cs_new_protected:Npn \__enumext_parse_foreach_keys:n #1
5788    {
5789      \exp_args:NV \__enumext_parse_foreach_keys:nn \l_keys_key_str {#1}
5790    }
```

We create the command.

```
5791  \NewDocumentCommand \foreachkeyans { +O{} m }
5792    {
5793      \__enumext_foreach_keyans:nn {#1} {#2}
5794    }
```

Finally the internal functions __enumext_foreach_keyans:nn and __enumext_foreach_add_body:n
will loop through the prop list and print the contents.

```
5795  \cs_new_protected:Npn \__enumext_foreach_keyans:nn #1 #2
5796    {
5797      \tl_use:N \l__enumext_foreach_default_keys_tl
5798      \keys_set:nn { enumext / foreach } {#1}
5799      \tl_set:Nn \l__enumext_foreach_name_prop_tl {#2}
5800      \prop_if_exist:cF { g__enumext_#2_prop }
5801        {
5802          \msg_error:nnn { enumext } { undefined-storage-anskey } {#2}
5803        }
5804      \int_compare:nNnT { \l__enumext_foreach_stop_int } = { 0 }
5805        {
5806          \int_set:Nn \l__enumext_foreach_stop_int
5807            { \prop_count:c { g__enumext_#2_prop } }
5808        }
5809      \seq_clear:N \l__enumext_foreach_print_seq
5810      \int_step_function:nnnN
5811        { \l__enumext_foreach_start_int }
5812        { \l__enumext_foreach_step_int }
5813        { \l__enumext_foreach_stop_int }
5814        \__enumext_foreach_add_body:n
5815      \seq_use:NV \l__enumext_foreach_print_seq \l__enumext_foreach_sep_tl
5816    }
5817  \cs_new_protected:Npn \__enumext_foreach_add_body:n #1
5818    {
5819      \seq_put_right:Ne \l__enumext_foreach_print_seq
5820        {
5821          \exp_not:V \l__enumext_foreach_before_tl
5822          \__enumext_foreach_wrapper:n
5823            {
5824              \prop_item:cn { g__enumext_ \l__enumext_foreach_name_prop_tl _prop }{#1}
5825            }
5826          \exp_not:V \l__enumext_foreach_after_tl
5827        }
5828    }
```

(*End of definition for* \foreachkeyans *and others. This function is documented on page* 19.)

13.54   Messages

Message used by package-load for multicol and hyperref packages.

```
5829  \msg_new:nnn { enumext } { package-load }
5830    {
5831      The~'#1'~package~is~already~loaded.
5832    }
5833  \msg_new:nnn { enumext } { package-not-load }
5834    {
5835      The~'#1'~package~will~be~loaded~as~a~dependency.
5836    }
```

Message used in the creation of counters by enumext package.

```
5837  \msg_new:nnn { enumext } { counters }
5838    {
5839      The~counter~'#1'~is~already~defined~by~some~\\
5840      package~or~macro,~it~cannot~be~continued.
5841    }
```

Message used by align and mark-pos keys.

```
5842  \msg_new:nnn { enumext } { unknown-choice }
5843    {
5844      The~value~'#3'~for~'#1'~key~is~invalid~use~('#2').
5845    }
```

Message used by reserved anskey* environment by enumext package.

```
5846 \msg_new:nnnn { enumext } { anskey-env-error }
5847   {
5848     The~environment~'#1'~is~reserved~by ~\\
5849     'enumext'~package,~It~is~already~defined.
5850   }
5851   {
5852     The~environment~'#1'~is~defined~internally ~
5853     for~the~'save-ans'~key~with~save-ans~key~active.~See~documentation.\\
5854   }
```

Message used in the creation of *prop list* by enumext package.

```
5855 \msg_new:nnn { enumext } { store-prop }
5856   {
5857     *~Package~enumext:~Creating ~
5858       \c_backslash_str g__enumext_#1_prop~\msg_line_context:.
5859   }
5860 \msg_new:nnn { enumext } { store-seq }
5861   {
5862     *~Package~enumext:~Creating ~
5863     \c_backslash_str g__enumext_#1_seq~\msg_line_context:.
5864   }
5865 \msg_new:nnn { enumext } { store-int }
5866   {
5867     *~Package~enumext:~Creating ~
5868     \c_backslash_str g__enumext_resume_#1_int~\msg_line_context:.
5869   }
5870 \msg_new:nnn { enumext } { prop-seq-int-hook }
5871   {
5872     *~Package~enumext:~Elements~in ~
5873     \c_backslash_str g__enumext_#1_prop~=~#2.\\
5874     *~Package~enumext:~Elements~in ~
5875     \c_backslash_str g__enumext_#1_seq~=~#3.\\
5876     *~Package~enumext:~Value~off ~
5877     \c_backslash_str g__enumext_resume_#1_int~=~#4.
5878   }
5879 \msg_new:nnn { enumext } { item-answer-hook }
5880   {
5881     *~Package~enumext:~Value~off ~
5882     \c_backslash_str g__enumext_item_number_int~=~#1.\\
5883     *~Package~enumext:~Value~off ~
5884     \c_backslash_str g__enumext_item_anskey_int~=~#2.\\
5885     *~Package~enumext:~Difference~item_number_int~-~item_anskey_int~=~#3.
5886   }
```

Message used by [⟨*key = val*⟩] system and \setenumext command.

```
5887 \msg_new:nnn { enumext } { invalid-key }
5888   {
5889     The~key~'#1'~is~not~know~the~level~#2.
5890   }
5891 \msg_new:nnn { enumext } { unknown-key-family }
5892   {
5893     Unknown~key~family~`\l_keys_key_str'~for~enumext.
5894   }
```

Messages used in length calculation.

```
5895 \msg_new:nnn { enumext } { width-negative }
5896   {
5897     Ignoring~negative~value~'#1=#2'~\msg_line_context:.\\
5898     The~key~'#1'~ accepts~values ~>=~0pt.
5899   }
```

Messages used by show-length key in enumext.

```
5900 \msg_new:nnn { enumext } { list-lengths }
5901   {
5902     ****~Lengths~used~by~'enumext'~level~'#2'~\msg_line_context:~\c_space_tl ****\\
5903     \__enumext_show_length:nnn { dim  } { labelsep     } {#1}
5904     \__enumext_show_length:nnn { dim  } { labelwidth   } {#1}
5905     \__enumext_show_length:nnn { dim  } { itemindent   } {#1}
5906     \__enumext_show_length:nnn { dim  } { leftmargin   } {#1}
5907     \__enumext_show_length:nnn { dim  } { rightmargin  } {#1}
5908     \__enumext_show_length:nnn { dim  } { listparindent } {#1}
5909     \__enumext_show_length:nnn { skip } { topsep      } {#1}
```

```
5910    \__enumext_show_length:nnn { skip } { parsep    } {#1}
5911    \__enumext_show_length:nnn { skip } { partopsep } {#1}
5912    \__enumext_show_length:nnn { skip } { itemsep   } {#1}
5913    ****************************************************
5914  }
```

Messages used by `show-length` key in `enumext*`, `keyans*` and `keyans`.

```
5915  \msg_new:nnn { enumext } { list-lengths-not-nested }
5916  {
5917    ****~Lengths~used~by~'#2'~environment~\msg_line_context:~\c_space_tl ****\\
5918    \__enumext_show_length:nnn { dim  } { labelsep     } {#1}
5919    \__enumext_show_length:nnn { dim  } { labelwidth   } {#1}
5920    \__enumext_show_length:nnn { dim  } { itemindent   } {#1}
5921    \__enumext_show_length:nnn { dim  } { leftmargin   } {#1}
5922    \__enumext_show_length:nnn { dim  } { rightmargin  } {#1}
5923    \__enumext_show_length:nnn { dim  } { listparindent } {#1}
5924    \__enumext_show_length:nnn { skip } { topsep    } {#1}
5925    \__enumext_show_length:nnn { skip } { parsep    } {#1}
5926    \__enumext_show_length:nnn { skip } { partopsep } {#1}
5927    \__enumext_show_length:nnn { skip } { itemsep   } {#1}
5928    ****************************************************
5929  }
```

Messages used by `ref` key.

```
5930  \msg_new:nnn { enumext } { key-ref-empty }
5931  {
5932    Key~'ref'~need~a~value~in~'#1'~ \msg_line_context:.
5933  }
```

Messages used by `save-ans` key.

```
5934  \msg_new:nnn { enumext } { save-ans-empty }
5935  {
5936    Key~'save-ans'~need~a~value~in~'#1'~ \msg_line_context:.
5937  }
5938  \msg_new:nnn { enumext } { save-ans-log }
5939  {
5940    *~Package~enumext:~Start~#1\c_space_tl with~save-ans=#2~\msg_line_context:.
5941  }
5942  \msg_new:nnn { enumext } { save-ans-log-hook }
5943  {
5944    *~Package~enumext:~Stop~#1\c_space_tl with~save-ans=#2~\msg_line_context:.
5945  }
```

Messages used by the internal system to check answer used by `check-ans` key.

```
5946  \msg_new:nnn { enumext } { items-same-answer }
5947  {
5948    ****************************************\\
5949    *~Package~enumext:~Checking~answers~in~'#1' ~
5950    for~\c_left_brace_str #2 \c_right_brace_str\\
5951    *~started~#3~and~close~\msg_line_context: : ~
5952    'OK',~all~items~with~answer.\\
5953    ****************************************
5954  }
5955  \msg_new:nnn { enumext } { item-greater-answer }
5956  {
5957    Checking~answers~in~'#1'~for~\c_left_brace_str #2 \c_right_brace_str\\
5958    started~#3~and~close~\msg_line_context: : ~'NOT~OK'\\
5959    Items~>~Answers.
5960  }
5961  \msg_new:nnn { enumext } { item-less-answer }
5962  {
5963    Checking~answers~in~'#1'~for~\c_left_brace_str #2 \c_right_brace_str\\
5964    started~#3~and~close~\msg_line_context: : ~'NOT~OK'\\
5965    Items~<~Answers.
5966  }
```

Messages used by the internal system to check for *"starred"* `\item*` and `\anspic*` commands.

```
5967  \msg_new:nnn { enumext } { missing-starred }
5968  {
5969    Missing~'\c_backslash_str #1*'~#2.
5970  }
5971  \msg_new:nnn { enumext } { many-starred }
5972  {
```

```
5973    Many~'\c_backslash_str #1*'~#2.
5974    }
```

Messages used by \printkeyans* command.

```
5975  \msg_new:nnn { enumext } { print-starred }
5976    {
5977      \c_backslash_str printkeyans*:~ The~sequence~'#1'~already~contains ~
5978      #2~environment~ \msg_line_context:.
5979    }
```

Message for the nesting depth of the environment enumext.

```
5980  \msg_new:nnn { enumext } { list-too-deep }
5981    {
5982      Too~deep~nesting ~for~'enumext'~\msg_line_context:.~ \\
5983      The~maximum ~level ~of ~nesting ~is~4.
5984    }
```

Messages used by \anskey, anskey* and \anspic commands.

```
5985  \msg_new:nnn { enumext } { anskey-unnumber-item }
5986    {
5987      Can't~store~with~a~unnumbered~\c_backslash_str item~\msg_line_context:.
5988    }
5989  \msg_new:nnn { enumext } { anskey-empty-arg }
5990    {
5991      Can't~store~empty~content~\msg_line_context:.
5992    }
5993  \msg_new:nnn { enumext } { anskey-wrong-place }
5994    {
5995      Wrong~place~for~command~'\c_backslash_str #1'~\msg_line_context:.~ \\
5996      '\c_backslash_str #1'~works~in~the~environment~'#2'.
5997    }
5998  \msg_new:nnn { enumext } { anskey-nested }
5999    {
6000      The~command~\c_backslash_str anskey~ can't~be~nested~\msg_line_context:.
6001    }
6002  \msg_new:nnn { enumext } { anskey-math-mode }
6003    {
6004      #1~can't~work~in~math~mode~\msg_line_context:.
6005    }
6006  \msg_new:nnn { enumext } { anskey-env-wrong }
6007    {
6008      The~environment~anskey*~cannot~use~in~'#1'~\msg_line_context:.
6009    }
6010  \msg_new:nnn { enumext } { command-wrong-place }
6011    {
6012      Wrong~place~for~command~'\c_backslash_str #1'~\msg_line_context:.~ \\
6013      '\c_backslash_str #1'~works~outside~the~environment~'#2'.
6014    }
6015  \msg_new:nnnn { enumext } { anskey-env-key-unknown }
6016    {
6017      The~key~'#1'~is~unknown~by~environment~
6018      'anskey*'~and~is~being~ignored.
6019    }
6020    {
6021      The~environment~'anskey*'~does~not~have~a~key~called ~'#1'.\\
6022      Check~that~you~have~spelled~the~key~name~correctly.
6023    }
6024  \msg_new:nnnn { enumext } { anskey-env-key-value-unknown }
6025    {
6026      The~key~'#1=#2'~is~unknown~by~environment ~
6027      'anskey*'~and~is~being~ignored.
6028    }
6029    {
6030      The~environment~'anskey*'~does~not~have~a~key~called ~'#1'.\\
6031      Check~that~you~have~spelled~the~key~name~correctly.
6032    }
6033  \msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
6034    { The~key~'#1'~is~unknown~by~'\c_backslash_str anskey'~and~is~being~ignored.}
6035    {
6036      The~command ~'\c_backslash_str anskey'~does~not~have~a~key~called ~'#1'.\\
6037      Check~that~you~have~spelled~the~key~name~correctly.
6038    }
```

```
6039  \msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
6040    { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str anskey'~and~is~being~ignored. }
6041    {
6042      The~command~'\c_backslash_str anskey'~does~not~have~a~key~called ~'#1'.\\
6043      Check~that~you~have~spelled~the~key~name~correctly.
6044    }
6045  \msg_new:nnn { enumext } { overwrite-file }
6046    {
6047      Overwriting~file~'#1'.
6048    }
6049  \msg_new:nnn { enumext } { writing-file }
6050    {
6051      Writing~file~'#1'.
6052    }
6053  \msg_new:nnn { enumext } { not-writing }
6054    {
6055      File~`#1'~already~exists.~Not~writing.
6056    }
```

Messages used by `keyans`, `keyans*` and `keyanspic` environment.

```
6057  \msg_new:nnn { enumext } { keyans-nested }
6058    {
6059      The~environment~'keyans'~can't~be ~nested ~\msg_line_context:.
6060    }
6061  \msg_new:nnn { enumext } { keyans-wrong-level }
6062    {
6063      Wrong~level~position~for~'keyans'~\msg_line_context:.~ \\
6064      The~environment~'keyans'~can~only~be~in~the~first~level.
6065    }
6066  \msg_new:nnn { enumext } { wrong-place }
6067    {
6068      Wrong~place~for~'#1'~environment ~\msg_line_context:.~ \\
6069      '#1'~is~only~found~with~'#2'~ in ~ 'enumext.
6070    }
6071  \msg_new:nnn { enumext } { keyanspic-nested }
6072    {
6073      The~environment~'keyanspic'~can't~be ~nested~ \msg_line_context:.~.
6074    }
6075  \msg_new:nnn { enumext } { keyanspic-wrong-level }
6076    {
6077      Wrong~level~position~for~'keyanspic'~\msg_line_context:.~ \\
6078      The~environment~'keyans'~can~only~be~in~the~first~level.
6079    }
6080  \msg_new:nnn { enumext } { keyanspic-item-cmd }
6081    {
6082      Can't~use ~\c_backslash_str item~in~keyanspic~\msg_line_context:.
6083    }
6084  \msg_new:nnnn { enumext } { keyans-unknown-key }
6085    {
6086      The~key~'#1'~is~unknown~by~environment~
6087      '\l__enumext_envir_name_tl'~and~is~being~ignored.
6088    }
6089    {
6090      The~environment~'\l__enumext_envir_name_tl'~does~not
6091    ~have~a~key~called ~'#1'.\\
6092      Check~that~you~have~spelled~the~key~name~correctly.
6093    }
6094  \msg_new:nnnn { enumext } { keyans-unknown-key-value }
6095    {
6096      The~key~'#1=#2'~is~unknown~by~environment ~
6097      '\l__enumext_envir_name_tl'~and~is~being~ignored.
6098    }
6099    {
6100      The~environment~'\l__enumext_envir_name_tl'~does~not
6101    ~have~a~key~called ~'#1'.\\
6102      Check~that~you~have~spelled~the~key~name~correctly.
6103    }
```

Message used by unknown ⟨*keys*⟩ in `enumext*`. environment.

```
6104  \msg_new:nnnn { enumext } { starred-unknown-key }
6105    {
6106      The~key~'#1'~is~unknown~by~environment~
```

```
6107      '\l__enumext_envir_name_tl'~and~is~being~ignored.
6108    }
6109    {
6110      The~environment~'\l__enumext_envir_name_tl'~does~not
6111    ~have~a~key~called ~'#1'.\\
6112      Check~that~you~have~spelled~the~key~name~correctly.
6113    }
6114 \msg_new:nnnn { enumext } { starred-unknown-key-value }
6115    {
6116      The~key~'#1=#2'~is~unknown~by~environment ~
6117      '\l__enumext_envir_name_tl'~and~is~being~ignored.
6118    }
6119    {
6120      The~environment~'\l__enumext_envir_name_tl'~does~not
6121    ~have~a~key~called ~'#1'.\\
6122      Check~that~you~have~spelled~the~key~name~correctly.
6123    }
```

Message used by unknown ⟨*keys*⟩ in enumext environment.

```
6124 \msg_new:nnnn { enumext } { standar-unknown-key }
6125    {
6126      The~key~'#1'~is~unknown~by~environment~'\l__enumext_envir_name_tl' \c_space_tl
6127    ~on~level~\int_use:N \l__enumext_level_int \c_space_tl and~is~being~ignored.
6128    }
6129    {
6130      The~environment~'\l__enumext_envir_name_tl'~does~not
6131    ~have~a~key~called ~'#1'~on~level~\int_use:N \l__enumext_level_int.\\
6132      Check~that~you~have~spelled~the~key~name~correctly.
6133    }
6134 \msg_new:nnnn { enumext } { standar-unknown-key-value }
6135    {
6136      The~key~'#1=#2'~is~unknown~by~environment~'\l__enumext_envir_name_tl' \c_space_tl
6137    ~on~level~\int_use:N \l__enumext_level_int \c_space_tl and~is~being~ignored.
6138    }
6139    {
6140      The~environment~'\l__enumext_envir_name_tl'~does~not
6141    ~have~a~key~called ~'#1'~on~level~\int_use:N \l__enumext_level_int.\\
6142      Check~that~you~have~spelled~the~key~name~correctly.
6143    }
```

Message used by unknown ⟨*keys*⟩ in \foreachkeyans.

```
6144 \msg_new:nnnn { enumext } { for-key-unknown }
6145    { The~key~'#1'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored.}
6146    {
6147      The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
6148      Check~that~you~have~spelled~the~key~name~correctly.
6149    }
6150 \msg_new:nnnn { enumext } { for-key-value-unknown }
6151    { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored. }
6152    {
6153      The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
6154      Check~that~you~have~spelled~the~key~name~correctly.
6155    }
```

Messages used by \getkeyans command.

```
6156 \msg_new:nnn { enumext } { undefined-storage-anskey }
6157    {
6158      Storage~named~'#1'~is~not~defined~\msg_line_context:.
6159    }
```

Messages used by \miniright command.

```
6160 \msg_new:nnn { enumext } { missing-miniright }
6161    {
6162      Missing~'\c_backslash_str miniright'~in~\msg_line_context:.\\
6163      The~key~'mini-env'~need~'\c_backslash_str miniright'.
6164    }
6165 \msg_new:nnn { enumext } { wrong-miniright-place }
6166    {
6167      Wrong~place~for~'\c_backslash_str miniright'~\msg_line_context:.~ \\
6168      Works~in~'enumext'~and~'keyans'~with~key~'mini-env'.
6169    }
6170 \msg_new:nnn { enumext } { wrong-miniright-use }
6171    {
```

```
6172        Wrong~use~for~'\c_backslash_str miniright'~\msg_line_context:.~ \\
6173        '\c_backslash_str miniright'~need~a~key~'mini-env'.
6174      }
6175  \msg_new:nnn { enumext } { wrong-miniright-starred }
6176      {
6177        Can't~use ~\c_backslash_str miniright~in~starred~environments~\msg_line_context:.
6178      }
6179  \msg_new:nnn { enumext } { many-miniright-used }
6180      {
6181        Can't~use ~\c_backslash_str miniright~more~than~once~ \msg_line_context:.
6182      }
```

Messages used by \setenumextmeta command.

```
6183  \msg_new:nnn { enumext } { already-defined }
6184      {
6185        The~key~'#1'~is~already~defined~\msg_line_context:.
6186      }
6187  \msg_new:nnn { enumext } { prohibited-unknown }
6188      {
6189        The~name~'unknown'~can't~be~chosen~ for~a~meta~key~\msg_line_context:.
6190      }
```

Messages used by enumext* and keyans* environments.

```
6191  \msg_new:nnn { enumext } { nested }
6192      {
6193        The~environment~\l__enumext_envir_name_tl \c_space_tl can't~be~nested~\msg_line_context:.
6194      }
6195  \msg_new:nnn { enumext } { nested-horizontal }
6196      {
6197        The~environment~\l__enumext_envir_name_tl \c_space_tl can't~be~nested~in~'#1'~ \msg_line_cont
6198      }
6199  \msg_new:nnn { enumext } { item-joined }
6200      {
6201        Items~joined~(#1)~>~#2 ~columns ~\msg_line_context:.
6202      }
6203  \msg_new:nnn { enumext } { item-joined-columns }
6204      {
6205        Not~space~to~join~items~(#1)~>~#2 ~\msg_line_context:.
6206      }
```

Messages used by resume key.

```
6207  \msg_new:nnn { enumext } { unknown-series-starred }
6208      {
6209        The~series~'#1'~for~the~resume~key~does~not~exist~in~the~
6210        ~enumext*~environment~ \msg_line_context:.
6211      }
6212  \msg_new:nnn { enumext } { unknown-series-standar }
6213      {
6214        The~series~'#1'~for~the~resume~key~does~not~exist~at~level~\int_use:N \l__enumext_level_int
6215        \c_space_tl of~enumext~environment~ \msg_line_context:.
6216      }
6217  \msg_new:nnnn { enumext } { out-of-range }
6218      { The~number~must~be~exactly~1,~2,~3~or~4. }
6219      { Received:~'#1'. }
```

13.55   Finish package

Finish package implementation.

```
6220  \file_input_stop:
6221  ⟨/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.