enumext
ENUMERATE EXERCISE SHEETS

V1.0   2024-09-18*

©2024 by Pablo González†

Abstract

This package provides *"enumerated list"* environments for creating *"simple exercise sheets"* along with *"multiple choice questions"*, storing the ⟨answers⟩ to these in memory using `multicol` and `scontents` packages and the `l3seq` and `l3prop` modules.

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 a note of the main ones:

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

---

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] and `scontents`[4] packages, need to have a modern TeX distribution such as TeX Live or MiKTeX. It has been tested with the standard classes provided by LaTeX: `book`, `report`, `article` and `letter` on `10pt`, `11pt` and `12pt`.

1 Introduction

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

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

1. Factor $x^2 - 2x + 1$
2. Factor $3x + 3y + 3z$
3. True False
   (a) $\alpha > \delta$
   (b) 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)

B) C)

5. Question with image on left side:

D) E)

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

Where what 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" *
*

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.

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

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 intend to replace the `enumerate` environment nor replace the powerful `enumitem`[6], the approach is intended to work without hindering either of them.

This package can be used with `xelatex`, `lualatex`, `pdflatex` and the classical `latex»dvips»ps2pdf` and is present in TEX Live and MiKTEX, use the package manager to install. For manual installation, download enumext.zip and unzip it, run `lualatex enumext.dtx` and move all files to appropriate locations, then run `mktexlsr`. To produce the documentation run `lualatex enumext.dtx` two times.

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

The package is loaded in the usual way:

```
\usepackage{enumext}
```

1.2 The concept of left margin

There is a direct relationship between the parameters `\leftmargin`, `\itemindent`, `\labelwidth` and `\labelsep` plus an *"extra space"* that makes it difficult to obtain the desired *horizontal spaces* in a `list` environment.

Usually we don't want the `list` to go beyond the left margin of the page, but since these four values are related, that causes a problem. The `enumitem`[6] package adds the `\labelindent` parameter to solve some of these problems. A simplified representation of this in the figure 1.

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

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

Figure 2: Representation of horizontal lengths concept in enumext.

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

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

1.3 User interface

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

1.3.1 Internal counters

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

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

1.3.2 Public dimension

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

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

1.3.3 Support for multicol

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

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

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

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

1.3.4 Support for `minipage`

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

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

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

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

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

🎯 The implementation of `\label` and `\ref` together with the `save-ref` key are compatible with the `hyperref`[8] package.

1.3.6 Support for \footnote

This package provides an internal implementation for the \footnote command which is compatible with the hyperref package for the enumext* and keyans* environments, but will not produce the expected links, and if the mini-env key is used in enumext or keyans environments the output will look like the classic way they are displayed in the environment minipage.

The best way to solve this is to use Jean-François Burnol footnotehyper[9] package, it will support keeping the links if hyperref is loaded with the hyperfootnotes=true option (default) and will show the output numbered at the bottom of the page (as opposed to how it is displayed in the minipage environment). The way to load it is as follows:

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

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 enumerate* environment provided by the enumitem package or task environment provided by the task package , \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"* in the environment is placed within a minipage environment whose *width* is stored in the dimension \itemwidth that NOT includes labelwith, labelsep, only the *width of the content.*

- You cannot have floating environments like figure or table but \footnote with hyperref support is supported if the footnotehyper package is loaded.

Example with columns=2

1. This text is in the first level.

X This text is in the first level.

2. This text is in the first level.

⋆ 4. This text is in the first level.

2.3 The command \item*

\item*

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

The \item*, \item*[⟨symbol⟩] and \item*[⟨symbol⟩][⟨offset⟩] works like the numbered \item, but placing a ⟨symbol⟩ to the *"left"* of the ⟨label⟩ separated from it by the ⟨offset⟩ set by the the second optional argument. The default values for ⟨symbol⟩ and ⟨offset⟩ are $\star$ '⋆' and the value set by labelsep key.

The *starred argument* '*' cannot be separated by spaces '␣' from the command, i.e. \item* and the first optional argument does *"not support"* verbatim content. Can be configure with the keys item-sym* and item-pos* locally in the environment or globally using \setenumext command (§3).

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

2.3.1 Keys for `\item*`

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

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

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

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

2.4 The command `\item` in `enumext*`

The `\item` command for the `enumext*` environment provides an optional *"first argument"* `\item(`⟨*columns*⟩`)` which *"joins items"* between columns. Let's consider the following examples adapted directly from the `task` package:

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

1. The first   ⋆ 2. The second   3. The third   4. The fourth

⋆ 5. The fifth item is way too long for this and needs three columns   6. The sixth

7. The seventh   X 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 arguments of environments and commands have the *highest precedence*, overriding both options passed by `\setenumext`. If the optional argument is not passed, the first level of the environment `enumext` will be taken by default.

🎯 The key `save-ans` that activate the *"storage system"* must NOT be passed through this command and must be passed directly in the optional argument of the *"first level"* of the environment in which they are executed.

4 The command `\setenumextmeta`

`\setenumextmeta`

$\setenumextmeta \{⟨key name⟩\}\{⟨key\text{-}one = val, key\text{-}two = val, ...⟩\}$
$\setenumextmeta*\{⟨key name⟩\}\{⟨key\text{-}one = val, key\text{-}two = val, ...⟩\}$
$\setenumextmeta [⟨enumext*⟩]\{⟨key name⟩\}\{⟨key\text{-}one = val, key\text{-}two = val, ...⟩\}$
$\setenumextmeta [⟨enumext, level⟩]\{⟨key name⟩\}\{⟨key\text{-}one = val, key\text{-}two = val, ...⟩\}$

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. If the optional argument is not passed, the new *"meta-key"* will be created for the first level of the environment `enumext`.

The starred version `*` will create the new *"meta-key"* for the environment `enumext*` and for all levels of the environment `enumext`.

5 The `keyval` system

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

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

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

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

5.1 Keys for `label` and `ref`

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

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

🛈 This key is intended to give the basic structure with which the $\langle label \rangle$ will be displayed, and the form in which it is used by standard *"label and ref"* and the *"internal reference"* system with the `save-ref` key. You cannot use commands with $\langle label \rangle$ as an argument, for example `\emph{`$\langle$`\alph*`$\rangle$`}` will return an error. For full customization of how $\langle label \rangle$ is displayed use the `font` or `wrap-label` keys.

`ref` = {⟨*code* {\alph*| \Alph*| \arabic*| \roman*| \Roman*} *more code*⟩} $\hfill$ 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{`$\langle$`\alph*`$\rangle$`}` 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`.

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

Sets the *horizontal space* between the box containing the current $\langle label \rangle$ defined by `label` key and the text of an item on the first line. Internally sets the value of `\labelsep` for the current level.

`labelwidth` = {⟨*rigid length*⟩} $\hfill$ default: *by label*

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

`widest` = {⟨*integer | string*⟩} $\hfill$ default: *empty*

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

`font` = {⟨*font commands*⟩} $\hfill$ default: *empty*

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

`align` = {⟨*left | right | center*⟩} $\hfill$ default: *left*

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

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

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

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

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

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

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

5.2 Keys for spaces

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

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

5.2.1 Vertical spaces

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

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

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

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

`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`.

`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 *"meta-key"* that does not receive an argument available only for the *first level* of environment `enumext` and environment `enumext*`. Fix the *baseline* when an environment `enumext` is nested in `enumext*` or vice versa 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`.

🫧 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 on the *"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 on the *"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"*.

5.2.2   Horizontal spaces

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

Extra *horizontal indentation*, beyond `labelsep`, of the *"first line"* off each item. This value is applied internally using `\hspace` and does not modify the value of `\itemindent`.

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.

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 environment `enumext` 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"*. This setting is equivalent (more or less) to the `wide` key provided by the `enumitem` package.

💣 For the `enumext*` and `keyans*` environments the keys `list-indent` and `list-offset` have the same effect.

5.3   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"* performing all calculations related to the *list parameters* in the environment and the parameters sets by `[`⟨*key = val*⟩`]` that is, in the second argument of the list after setting all the parameters `\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 `[`⟨*key = val*⟩`]` sets in the environment that is, before the arguments defining the environment are executed: {⟨*code*⟩}`\begin{list}{`⟨*arg one*⟩`}{`⟨*arg two*⟩`}`.

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

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

💣 Keep in mind that the code set in this key will affect the entire *"body"* of the environment and therefore the inner levels of the list and the `keyans` environment. It is recommended to set this key per level.

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

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

5.4   Keys for `start`, `series` and `resume`

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

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

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

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

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

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

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

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

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

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

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

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

5.5   Keys for multicols

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

Set the *number of columns* to be used by the multicols environment within the environment. The value must be a positive integer less than or equal to 10.

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

Set the *space between* columns used by the multicols environment within the environment. Internally sets the value of \columnsep, by default its value is equal to the sum of the values set in the keys labelwidth and labelsep of the current level.

🎯 The \footnote{⟨*text*⟩} command in the nested levels of multicols will not work as expected, prefer the use of \footnotemark[⟨*number*⟩] inside the environment and \footnotetext[⟨*number*⟩]{⟨*text*⟩} outside the environment or via the after key.

5.6   Keys for minipage

mini-env = {⟨*rigid length*⟩}   default: *not used*

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

mini-sep = {⟨*rigid length*⟩}   default: *0.3333em*

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

5.6.1   The command \miniright

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

The \miniright command close the minipage environment on the *"left side"* and opens the minipage environment on the *"right side"* by starting it with the \centering command. It must be placed *"after"* the last \item of the current environment and *"before"* starting the material to be placed on the *"right side"*.
The *starred argument* '*' inhibits the use of \centering command i.e. the usual LaTeX justification is maintained in the minipage on the *"right side"*.

🎯 The \footnote{⟨*text*⟩} command in minipage environment will work as usual. If you prefer the footnotes to be numbered (not lowercase) and outside the environment, use \footnotemark[⟨*number*⟩] inside the environment and \footnotetext[⟨*number*⟩]{⟨*text*⟩} outside the environment or via the after key (see §1.3.6 for full support).

5.6.2   The key mini-right

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

mini-right = {⟨*content*⟩}   default: *not used*

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

mini-right* = {⟨*content*⟩}   default: *not used*

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

6 The storage system

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

```
\begin{enumext}[save-ans={⟨store name⟩}]
    \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 arguments passed to the inner levels or the environment nested in it, along with the content passed to `\anskey`, the current ⟨*labels*⟩ for `\item*` and `\anspic*` in the environments `keyans`, `keyans*` and `keyanspic` will be stored in a ⟨*sequence*⟩ and at the same time will be stored (without the environment structure or optional arguments) in a ⟨*prop list*⟩.

The optional arguments of the inner levels or the nested environment are filtered by excluding all ⟨*keys*⟩ related to the *"stored system"* along with the keys `series`, `resume` and `resume*` when storing in ⟨*sequence*⟩.

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, `\item*` in `keyans` and `keyans*` environments and `\anspic*` in `keyanspic` environment. If the ⟨*sequence*⟩ or ⟨*prop list*⟩ 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 arguments of the inner levels or nested environment that will be passed to the ⟨*sequence*⟩. The ⟨*key list*⟩ passed to this key ignores any ⟨*keys*⟩ in the *"stored system"* and must be passed between braces. For example, if we execute at a second level:

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

The ⟨*keys*⟩ that will be stored by default in the ⟨*sequence*⟩ would be `nosep, columns=2`, but using the key `save-key={columns=3}` will overwrite this and store it in the ⟨*sequence*⟩ only the key `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 `keyans`, `keyans*` and `keyanspic` environments and storing them in the ⟨*store name*⟩ defined by the `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.

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 ⟨*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 ⟨*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` within the environment in which the key `save-ans=test` was set.

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

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

6.1.2 Keys for `wrap` and `display`

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

Wraps the *argument* passed to the `\anskey` and the *body* in `anskey*` environment referenced by `{#1}` when using the `show-ans` or `show-pos` keys. The `{⟨code⟩}` must be passed between braces and only affects the *argument* or *body* 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}`'.

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

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

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

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

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

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

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

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

6.1.3 Keys for debug and checking

`show-pos` = {⟨*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.

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

This is a *meta-key* that does not receive an argument and disables the structure stored in the *sequence* `{⟨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`, *"without"* use `anskey*`, *"without"* interfering with the `check-ans` key and *"without"* storing an unwanted structure in the *sequence* `{⟨store name⟩}`.

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 `{⟨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: *$\star$*

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

6.3.1 Keys for anskey*

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

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

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

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

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

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

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

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

Example

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

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

⋆ 5. Text containing our instructions or questions.

[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.

```
\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*).

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 ⟨*content*⟩ (if it is present) 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.

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

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}[⟨nº above, nº below⟩]\anspic{⟨drawing⟩}\anspic*[⟨content⟩]{⟨drawing⟩}

The keyanspic is a *"fake enumerated list"* environment that which uses the \anspic command instead of \item. It is activated by the save-ans key and has the same settings as the keyans environment. It is intended for placing *"drawings"* or *"tabular"* with an in-line or *above* and *below* layout. A representation of the output can be seen in the figure 6.

Figure 6: Representation of the keyanspic environment with optional argument [3,2] in enumext.

The optional argument determines the number drawings or tabular *"above"* and *"below"* within the environment. The vertical separation between *"above"* and *"below"* is controlled by the values set by parsep and itemsep keys passed to keyans environment. If the optional argument or the second part of it is omitted the drawings or tabular will be put on a single line.

6.5.1   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 ⟨content⟩ (if it is present) in *sequence* and *prop list* {⟨store name⟩} set by save-ans key.

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

Example

```
\begin{enumext}[save-ans=test,show-ans,nosep]
  \item Question with images.
    \begin{keyanspic}[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}
\end{enumext}
```

1. Question with images.

A

B

A

A) B) A

A

C)

6.6 Printing stored content D)

* E)[note]

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: *empty*
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 increment (⟨*step*⟩) applied to the value set by key start for each element 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 it will finish executing. 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[⟨*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 environment or the 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=2,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. 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
   (A) value
   (B) correct
   (C) value
   (D) value

2. Second type of questions
   I. $2\alpha + 2\delta = 90°$

---

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

II. $\alpha = \delta$

III. $\angle EDF = 45°$

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

3. Third type of questions

   (1) $2\alpha + 2\delta = 90°$
   (2) $\angle EDF = 45°$

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

4. Question with image and label below:

5. Question with image on left side:

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

Test keys

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

Example 4

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

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)

8 The way of non-enumerated lists

It is possible to use (or abuse) the `enumext` environment to mimic *non-enumerated* list environments such as `itemize` and `description`, clearly the ⟨*keys*⟩ to *"store answers"*, the `keyans` and `keyanspic` environments lose their sense and it is not the focus of the main of this 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=2.5`em`,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=0`pt 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.

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=4`pt`,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*` key 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 }
  {%
    \IfBooleanTF{#1}
      {\strut\smash{\parbox[t]{\labelwidth}{\raggedright{#2}}}}%
      {\strut\smash{\parbox[t]{\labelwidth}{\raggedleft{#2}}}}%
  }
```

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

9   References

[1] HIRSCHHORN, PHILIP. "Using the exam document class". Available from CTAN, `https://www.ctan.org/pkg/exam`, 2023.

[2] NIEDERBERGER, CLEMENS. "xsim – eXercise Sheets IMproved". Available from CTAN, `https://www.ctan.org/pkg/xsim`, 2023.

[3] MITTELBACH, FRANK. "An environment for multicolumn output". Available from CTAN, `https://www.ctan.org/pkg/multicol`, 2024.

[4] GONZÁLEZ, PABLO. "scontents - Stores LaTeX contents in memory or files". Available from CTAN, `https://www.ctan.org/pkg/scontents`, 2022.

[5] The LaTeX Project. "enumerate – Enumerate with redefinable labels". Available from CTAN, `https://www.ctan.org/pkg/enumerate`, 2024.

[6] BEZOS, JAVIER. "Customizing lists with the enumitem package". Available from CTAN, `https://www.ctan.org/pkg/enumitem`, 2019

[7] BERRY, KARL. "LaTeX 2_ε: An Unofficial Reference Manual". Available from CTAN, `https://ctan.org/pkg/latex2e-help-texinfo`, 2024.

[8] The LaTeX Project. "Extensive support for hypertext in LaTeX". Available from CTAN, `https://www.ctan.org/pkg/hyperref`, 2024.

[9] BURNOL, JEAN-FRANÇOIS. "The footnotehyper package". Available from CTAN, `https://www.ctan.org/pkg/footnotehyper`, 2021.

[10] The LaTeX Project. "The expl3 package". Available from CTAN, `https://www.ctan.org/pkg/l3kernel`, 2024.

[11] The LaTeX Project. "The LaTeX3 Interfaces". Available from CTAN, `https://www.ctan.org/pkg/l3kernel`, 2024.

[12] The LaTeX Project. "The LaTeX 2_ε sources". Available from CTAN, `https://ctan.org/tex-archive/macros/latex/base`, 2024.

[13] The LaTeX Project. "LaTeX for authors current version". Available from CTAN, `https://ctan.org/pkg/latex-base`, 2024.

[14] GUNDLACH, PATRICK. "The lua-visual-debug package". Available from CTAN, `https://www.ctan.org/pkg/lua-visual-debug`, 2023.

[15] LEMVIG, MOGENS. "The shortlst package". Available from CTAN, `https://www.ctan.org/pkg/shortlst`, 1998.

[16] NIEDERBERGER, CLEMENS. "tasks – Horizontally columned lists". Available from CTAN, `https://www.ctan.org/pkg/tasks`, 2022.

10   Change history

v1.0   2024-09-18   – First public release.

11   Index of Documentation

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

12   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.

12.1   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.

12.2   Initial set up

Start the DocStrip guards.

```
1 ⟨*package⟩
```

Identify the internal prefix (LaTeX3 DocStrip convention) for l3doc class.

```
2 ⟨@@=enumext⟩
```

12.3   Declaration of the package

First we will make sure we have a minimum (super updated) version of LaTeX to work correctly.

```
3 \NeedsTeXFormat{LaTeX2e}[2024-06-01]
```

Now declare the enumext package.

```
4 \ProvidesExplPackage
5   {enumext}
6   {2024-09-18}
7   {1.0}
8   {Enumerate exercise sheets}
```

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

```
9  \hook_gput_code:nnn {begindocument} {enumext}
10   {
11     \IfPackageLoadedTF { multicol }
12       {
13         \msg_info:nnn { enumext } { package-load } { multicol }
14       }
15       {
16         \msg_info:nnn { enumext } { package-not-load } { multicol }
17         \RequirePackage{multicol}[2024-05-23]
18       }
19     \IfPackageLoadedTF { scontents }
20       {
21         \msg_info:nnn { enumext } { package-load } { scontents }
22       }
23       {
24         \msg_info:nnn { enumext } { package-not-load } { scontents }
25         \RequirePackage{scontents}
26       }
27   }
```

12.4   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 and `\anskey` command.

```
28 \int_new:N  \l__enumext_level_int
29 \int_new:N  \l__enumext_level_h_int
30 \int_new:N  \l__enumext_anskey_level_int
31 \int_new:N  \l__enumext_keyans_level_int
32 \int_new:N  \l__enumext_keyans_level_h_int
33 \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_anskey_env_bool`
`\l__enumext_keyans_env_bool`
`\g__enumext_start_line_tl`
`\g__enumext_envir_name_tl`
`\l__enumext_envir_name_tl`

Internal variables used by functions `\__enumext_is_not_nested:`, `\__enumext_is_on_first_level:` and `\__enumext_keyans_name_and_start:` (§12.5.1).

```
34 \bool_new:N \l__enumext_starred_bool
35 \bool_new:N \g__enumext_starred_bool
36 \bool_new:N \l__enumext_starred_first_bool
37 \bool_new:N \l__enumext_standar_bool
38 \bool_new:N \g__enumext_standar_bool
39 \bool_new:N \l__enumext_standar_first_bool
40 \bool_new:N \l__enumext_anskey_env_bool
41 \bool_new:N \l__enumext_keyans_env_bool
42 \tl_new:N    \g__enumext_start_line_tl
43 \tl_new:N    \g__enumext_envir_name_tl
44 \tl_new:N    \l__enumext_envir_name_tl
```

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

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

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

The initial values of these variables are set by the function `\__enumext_define_counters:Nn` (§12.10) and then modified by the function `\__enumext_label_style:Nnn` used by label key (§12.13).

```
45 \cs_set_protected:Npn \__enumext_tmp:n #1
46   {
47     \tl_new:c { l__enumext_counter_#1_tl }
48   }
49 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }
```

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

`\c__enumext_counter_style_tl`
`\l__enumext_ref_key_arg_tl`
`\l__enumext_ref_the_count_tl`
`\l__enumext_the_counter_X_tl`
`\l__enumext_renew_the_count_X_tl`

Internal variables used by ref key (§12.13).

```
50 \tl_const:Nn \c__enumext_counter_style_tl
51   { { arabic } { roman } { Roman } { alph } { Alph } }
52 \tl_new:N \l__enumext_ref_key_arg_tl
53 \tl_new:N \l__enumext_ref_the_count_tl
54 \cs_set_protected:Npn \__enumext_tmp:n #1
55   {
56     \tl_new:c  { l__enumext_renew_the_count_#1_tl }
57     \tl_new:c  { l__enumext_the_counter_#1_tl }
58     \tl_set:ce { l__enumext_the_counter_#1_tl } { \exp_not:c { theenumX#1 } }
59   }
60 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }
```

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

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

Internal variables used by resume, resume* and series keys (§12.24).

```
61 \int_new:N  \g__enumext_resume_int
62 \int_new:N  \g__enumext_resume_vii_int
63 \tl_new:N    \l__enumext_resume_name_tl
64 \bool_new:N \l__enumext_resume_active_bool
65 \tl_new:N    \g__enumext_standar_series_tl
66 \tl_new:N    \g__enumext_starred_series_tl
```

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

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

The variable `\l__enumext_current_widest_dim` stores the current label width, the variable `\g__enumext_counter_styles_tl` stores the default ⟨*label style*⟩ and the variable `\g__enumext_widest_label_tl` the label width. These variables are used by widest (§12.14) and label (§12.12) keys.

```
67 \dim_new:N \l__enumext_current_widest_dim
68 \tl_new:N  \g__enumext_counter_styles_tl
69 \tl_new:N  \g__enumext_widest_label_tl
70 \box_new:N \l__enumext_label_width_by_box
```

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

\l__enumext_leftmargin_tmp_X_bool
\l__enumext_leftmargin_tmp_X_dim
\l__enumext_leftmargin_X_dim
\l__enumext_itemindent_X_dim

The boolean variable \l__enumext_leftmargin_tmp_X_bool and the dimensional variable \l__enumext_-leftmargin_tmp_X_dim are used by the list-indent key (§12.17). The variables \l__enumext_-leftmargin_X_dim and \l__enumext_itemindent_X_dim are used and set by the function __enumext_-calc_hspace:NNNNNNNNNNNN (§12.37.1).

```
71 \cs_set_protected:Npn \__enumext_tmp:n #1
72   {
73     \bool_new:c { l__enumext_leftmargin_tmp_#1_bool }
74     \dim_new:c  { l__enumext_leftmargin_tmp_#1_dim }
75     \dim_new:c  { l__enumext_leftmargin_#1_dim      }
76     \dim_new:c  { l__enumext_itemindent_#1_dim      }
77   }
78 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }
```

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

\l__enumext_multicols_above_X_skip
\l__enumext_multicols_below_X_skip
\g__enumext_multicols_right_X_skip

Internal variables used by columns key §12.21).

```
79 \cs_set_protected:Npn \__enumext_tmp:n #1
80   {
81     \skip_new:c  { l__enumext_multicols_above_#1_skip }
82     \skip_new:c  { l__enumext_multicols_below_#1_skip }
83     \skip_new:c  { g__enumext_multicols_right_#1_skip }
84   }
85 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }
```

(*End of definition for* \l__enumext_multicols_above_X_skip, \l__enumext_multicols_below_X_skip, *and* \g__enumext_-multicols_right_X_skip.)

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

Internal variables used by \miniright command (§12.22.4) and the keys mini-right, mini-right*, mini-env and mini-sep (§12.20, §12.22).

```
86 \int_new:N  \g__enumext_minipage_stat_int
87 \skip_new:N \l__enumext_minipage_left_skip
88 \skip_new:N \l__enumext_minipage_right_skip
89 \skip_new:N \l__enumext_minipage_after_skip
90 \skip_new:N \g__enumext_minipage_right_skip
91 \skip_new:N \g__enumext_minipage_after_skip
92 \cs_set_protected:Npn \__enumext_tmp:n #1
93   {
94     \dim_new:c  { l__enumext_minipage_left_#1_dim     }
95     \bool_new:c { l__enumext_minipage_active_#1_bool }
96   }
97 \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }
```

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

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

The bool vars \l__enumext_wrap_label_X_bool and \l__enumext_wrap_label_opt_X_bool are used by wrap-label and wrap-label* keys (§12.12), the integer \l__enumext_start_X_int are used by the start and start* keys (§12.14), the token list \l__enumext_fake_item_indent_X_tl is used by itemindent key (§12.17.1), the variables \l__enumext_label_fill_left_X_tl and \l__enumext_-label_fill_left_X_tl are used by the align key (§12.12). 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 (§12.19).

```
98 \cs_set_protected:Npn \__enumext_tmp:n #1
99   {
100     \bool_new:c { l__enumext_wrap_label_#1_bool     }
101     \bool_new:c { l__enumext_wrap_label_opt_#1_bool }
102     \int_new:c  { l__enumext_start_#1_int           }
103     \tl_new:c   { l__enumext_fake_item_indent_#1_tl }
104     \tl_new:c   { l__enumext_label_fill_left_#1_tl  }
105     \tl_new:c   { l__enumext_label_fill_right_#1_tl }
106     \bool_new:c { l__enumext_vspace_a_star_#1_bool  }
107     \bool_new:c { l__enumext_vspace_b_star_#1_bool  }
108   }
109 \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.*)

The variable `\l__enumext_store_active_bool` setting by save-ans key (§12.25.1) activates all the mechanism related to `\anskey`, anskey*, keyans, keyans* and keyanspic environments.

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

The variable `\l__enumext_store_anskey_arg_tl` save the *argument* of `\anskey` (§12.29) and the variables `\l__enumext_store_anskey_env_tl` and `\l__enumext_store_anskey_opt_tl` save the ⟨*body*⟩ and the ⟨*keys*⟩ of the environment anskey* (§12.30).

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*` (§12.36) and `\anspic*` (§12.41.2) for the keyans, keyans* and keyanspic environments.

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

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

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

Internal variables used by the command `\setenumext` (§12.47).

```
119 \tl_new:N  \l__enumext_setkey_tmpa_tl
120 \tl_new:N  \l__enumext_setkey_tmpb_tl
121 \int_new:N \l__enumext_setkey_tmpa_int
122 \seq_new:N \l__enumext_setkey_tmpa_seq
123 \seq_new:N \l__enumext_setkey_tmpb_seq
```

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

Internal variables used by the `\printkeyans` command (§12.46) and `\foreachkeyans` command (§12.49).

```
124 \tl_new:N  \l__enumext_meta_path_tl
125 \seq_new:N \l__enumext_foreach_print_seq
126 \tl_new:N  \l__enumext_foreach_name_prop_tl
127 \tl_new:N  \g__enumext_foreach_default_keys_tl
```

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

Internal variables used by command `\printkeyans` (§12.46), show-pos key (§12.26), item-sym* key (§12.34), save-key key (§12.26.2) and *"storage level system"*.

```
128 \tl_new:N  \l__enumext_print_keyans_starred_tl
129 \str_new:N \l__enumext_mark_position_str
130 \tl_new:N  \g__enumext_item_symbol_aux_tl
131 \cs_set_protected:Npn \__enumext_tmp:n #1
132   {
133     \tl_new:c   { l__enumext_print_keyans_#1_tl        }
134     \tl_new:c   { l__enumext_store_save_key_#1_tl      }
135     \bool_new:c { l__enumext_store_save_key_#1_bool    }
136     \bool_new:c { l__enumext_store_upper_level_#1_bool }
137   }
138 \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.*)

Internal variables used by keyanspic environment and `\anspic` command (§12.41.1).

```
139 \seq_new:N  \l__enumext_keyans_pic_body_seq
140 \dim_new:N  \l__enumext_keyans_pic_width_dim
141 \int_new:N  \l__enumext_keyans_pic_above_int
142 \int_new:N  \l__enumext_keyans_pic_below_int
143 \skip_new:N \l__enumext_keyans_pic_above_skip
144 \bool_new:N \l__enumext_keyans_pic_star_bool
145 \str_new:N  \l__enumext_keyans_pic_label_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_keyans_pic_body_seq *and others.*)

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

Internal variables used by *"internal check answer"* mechanism (§12.25.3) used by the check-ans and no-store keys and check for starred commands \item* in keyans and keyans* environments and \anspic* in keyanspic environment.

```
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 \int_new:N  \g__enumext_check_starred_cmd_int
154 \int_new:N  \g__enumext_item_anskey_int
155 \int_new:N  \g__enumext_item_number_int
156 \bool_new:N \l__enumext_item_number_bool
157 \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 (§12.8). The boolean variable \l__enumext_footnotes_key_bool determine if hyperref is load with key hyperfootnotes=true.

```
158 \bool_new:N \l__enumext_hyperref_bool
159 \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 (§12.26). 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 (§12.8) and the variable \l__enumext_write_aux_file_tl will be in charge of executing the writing code in the .aux file.

```
160 \tl_new:N \l__enumext_newlabel_arg_one_tl
161 \tl_new:N \l__enumext_newlabel_arg_two_tl
162 \tl_new:N \l__enumext_write_aux_file_tl
163 \cs_set_protected:Npn \__enumext_tmp:n #1
164   {
165     \tl_new:c { l__enumext_label_copy_#1_tl }
166   }
167 \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_int
\g__enumext_footnote_arg_seq
\g__enumext_footnote_int_seq

Internal variables used for redefinition of \footnote (§12.42.1).

```
168 \int_new:N \g__enumext_footnote_int
169 \seq_new:N \g__enumext_footnote_arg_seq
170 \seq_new:N \g__enumext_footnote_int_seq
```

(*End of definition for* \g__enumext_footnote_int, \g__enumext_footnote_arg_seq, *and* \g__enumext_footnote_int_seq*.*)

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

Internal variables used by enumext* and keyans* environments.

```
171 \cs_set_protected:Npn \__enumext_tmp:n #1
172   {
173     \bool_new:c { l__enumext_item_starred_#1_bool      }
174     \int_new:c  { l__enumext_item_column_pos_#1_int    }
175     \int_new:c  { g__enumext_item_count_all_#1_int     }
176     \int_new:c  { l__enumext_joined_item_#1_int        }
177     \int_new:c  { l__enumext_joined_item_aux_#1_int    }
178     \int_new:c  { l__enumext_tmpa_#1_int               }
179     \dim_new:c  { l__enumext_tmpa_#1_dim               }
180     \box_new:c  { l__enumext_item_text_#1_box          }
181     \dim_new:c  { l__enumext_joined_width_#1_dim       }
182     \dim_new:c  { l__enumext_item_width_#1_dim         }
183     \tl_new:c   { g__enumext_item_symbol_aux_#1_tl     }
184     \str_new:c  { l__enumext_align_label_#1_str        }
185     \bool_new:c { g__enumext_minipage_active_#1_bool }
186     \box_new:c  { l__enumext_miniright_code_#1_box   }
187     \bool_new:c { g__enumext_minipage_center_#1_bool }
188     \dim_new:c  { g__enumext_minipage_right_#1_dim   }
189     \skip_new:c { g__enumext_minipage_right_#1_skip  }
190   }
191 \clist_map_inline:nn { vii, viii } { \__enumext_tmp:n {#1} }
```

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

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

```
192 \clist_const:Nn \c__enumext_all_envs_clist
193   {
194     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
195     {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
196   }
```

(*End of definition for* \c__enumext_all_envs_clist.)

12.5 Some utility functions

\keys_precompile:neN Non-standard kernel variants used by the \printkeyans command (§12.46) and \foreachkeyans command
\seq_use:NV (§12.49).

```
197 \cs_generate_variant:Nn \keys_precompile:nnN { neN }
198 \cs_generate_variant:Nn \seq_use:Nn { NV }
```

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

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

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

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

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

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

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

__enumext_level: Function for check current level in enumext.

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

(*End of definition for* __enumext_level:.)

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

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

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

__enumext_regex_counter_style: The internal function __enumext_regex_counter_style: replace the '*' with the actual counter of the running level and is used by the ref key. It loops through the defined counter styles in \c__enumext_-counter_style_tl and replace '*' by real command, for example, looking for \arabic* and replacing that by \arabic{⟨counter⟩} defined on the current level.

```
221 \cs_new_protected:Nn \__enumext_regex_counter_style:
222   {
223     \tl_map_inline:Nn \c__enumext_counter_style_tl
224       {
225         \regex_replace_once:nnN { \c{##1}\* }
226           { \c{##1}\cB{\u{l__enumext_ref_the_count_tl}\cE} } \l__enumext_ref_key_arg_tl
227       }
228   }
```

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

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

```
229  \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
230    {
231      * ~ #2
232      \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
233        = ~ \use:c { #1_use:c } { l__enumext_#2_#3_#1 } \\
234    }
```

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

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

```
235  \cs_new_protected:Npn \__enumext_unskip_unkern:
236    {
237      \int_case:nnT { \lastnodetype }
238        {
239          { 11 }
240            {
241              \typeout{SKIIIIIIIIIIIIIIIIP}
242              \typeout{\the\lastskip}
243              \unskip
244            }
245          { 12 }
246            {
247              \typeout{KERRRRRRRRRRRRRRRRN}
248              \typeout{\the\lastkern}
249              \unkern
250            }
251        }
252    }
```

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

12.5.1   Utilities for environments and levels

`\__enumext_is_not_nested:`
`\__enumext_is_on_first_level:`
The function `\__enumext_is_not_nested:` set the variables `\g__enumext_standar_bool` and `\g__enumext_starred_bool` to *"true"* only if the environments enumext and enumext* are nested in each other and save the environment name in `\l__enumext_envir_name_tl`.

```
253  \cs_new_protected:Nn \__enumext_is_not_nested:
254    {
255      \str_case:en { \@currenvir }
256        {
257          {enumext}
258            {
259              \tl_set:Nn \l__enumext_envir_name_tl { enumext }
260              \bool_lazy_and:nnT
261                { \bool_not_p:n { \g__enumext_standar_bool } }
262                { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
263                {
264                  \bool_gset_true:N \g__enumext_standar_bool
265                }
266            }
267          {enumext*}
268            {
269              \tl_set:Nn \l__enumext_envir_name_tl { enumext* }
270              \bool_lazy_and:nnT
271                { \bool_not_p:n { \g__enumext_starred_bool } }
272                { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
273                {
274                  \bool_gset_true:N \g__enumext_starred_bool
275                }
276            }
277        }
278    }
```

The function `\__enumext_is_on_first_level:` will set the variables `\l__enumext_standar_first_-bool` (§12.25.1), `\l__enumext_starred_first_bool` (§12.25.1) and `\l__enumext_anskey_env_bool` (§12.30) 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.

```
279  \cs_new_protected:Nn \__enumext_is_on_first_level:
280    {
281      \bool_lazy_all:nT
282        {
283          { \bool_if_p:N \g__enumext_standar_bool }
284          { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
285          { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
286        }
287        {
288          \bool_set_true:N \l__enumext_standar_first_bool
289          \bool_set_true:N \l__enumext_anskey_env_bool
290          \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
291          \tl_gset:Ne \g__enumext_start_line_tl
292            {
293              on ~ line ~ \exp_not:V \inputlineno
294            }
295        }
296      \bool_lazy_all:nT
297        {
298          { \bool_if_p:N \g__enumext_starred_bool }
299          { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
300          { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
301        }
302        {
303          \bool_set_true:N \l__enumext_starred_first_bool
304          \bool_set_true:N \l__enumext_anskey_env_bool
305          \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
306          \tl_gset:Ne \g__enumext_start_line_tl
307            {
308              on ~ line ~ \exp_not:V \inputlineno
309            }
310        }
311    }
```

(*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.

```
312  \cs_new_protected:Nn \__enumext_keyans_name_and_start:
313    {
314      \str_case:en { \@currenvir }
315        {
316          {keyans}
317            {
318              \tl_set:Nn \l__enumext_envir_name_tl { keyans }
319              \tl_set:Ne \l__enumext_check_start_line_env_tl
320                {
321                  in ~ 'keyans' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
322                }
323            }
324          {keyans*}
325            {
326              \tl_set:Nn \l__enumext_envir_name_tl { keyans* }
327              \tl_set:Ne \l__enumext_check_start_line_env_tl
328                {
329                  in ~ 'keyans*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
330                }
331            }
332          {keyanspic}
333            {
334              \tl_set:Nn \l__enumext_envir_name_tl { keyanspic }
335              \tl_set:Ne \l__enumext_check_start_line_env_tl
336                {
337                  in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
338                }
339            }
340        }
341    }
```

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

12.5.2 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.

```
342 \cs_new_protected:Nn \__enumext_reset_global_vars:
343   {
344     \__enumext_reset_global_int:
345     \__enumext_reset_global_bool:
346     \__enumext_reset_global_tl:
347   }
348 \cs_new_protected:Nn \__enumext_reset_global_int:
349   {
350     \int_gzero:N \g__enumext_item_number_int
351     \int_gzero:N \g__enumext_item_anskey_int
352     \int_gzero:N \g__enumext_item_answer_diff_int
353   }
354 \cs_new_protected:Nn \__enumext_reset_global_bool:
355   {
356     \bool_gset_false:N \g__enumext_check_ans_key_bool
357     \bool_gset_false:N \g__enumext_standar_bool
358     \bool_gset_false:N \g__enumext_starred_bool
359   }
360 \cs_new_protected:Nn \__enumext_reset_global_tl:
361   {
362     \tl_gclear:N \g__enumext_store_name_tl
363     \tl_gclear:N \g__enumext_start_line_tl
364     \tl_gclear:N \g__enumext_envir_name_tl
365   }
```

(*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.

```
366 \cs_new_protected:Nn \__enumext_log_global_vars:
367   {
368     \msg_log:nneeee { enumext } { prop-seq-int-hook }
369       { \g__enumext_store_name_tl }
370       { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
371       { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
372       { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
373   }
```

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.

```
374 \cs_new_protected:Nn \__enumext_log_answer_vars:
375   {
376     \msg_log:nneee { enumext } { item-answer-hook }
377       { \int_use:N \g__enumext_item_number_int }
378       { \int_use:N \g__enumext_item_anskey_int }
379       { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int} }
380   }
```

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

12.6 Copying list and minipage environments

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

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

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

`\__enumext_start_list:nn`
`\__enumext_stop_list:`
`\__enumext_item_std:w`

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

```
381 \__enumext_at_begin_document:n
382   {
```

```
383    \cs_new_eq:NN   \__enumext_start_list:nn \list
384    \cs_new_eq:NN   \__enumext_stop_list: \endlist
385    \NewCommandCopy \__enumext_item_std:w \item
386  }
```

(*End of definition for* __enumext_start_list:nn, __enumext_stop_list:, *and* __enumext_item_std:w.)

The 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 minipage environment or a related command.

__enumext_minipage:w
__enumext_endminipage:

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

```
387 \__enumext_at_begin_document:n
388   {
389      \cs_new_eq:NN \__enumext_minipage:w \minipage
390      \cs_new_eq:NN \__enumext_endminipage: \endminipage
391   }
```

(*End of definition for* __enumext_minipage:w *and* __enumext_endminipage:.)

12.7   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. This function is passed to the function __enumext_safe_exec: in the enumext environment definition (§12.38) and __enumext_safe_exec_vii: in the enumext* environment definition (§12.43)

```
392 \cs_new_protected:Nn \__enumext_internal_mini_page:
393   {
394      \int_compare:nNnT { \l__enumext_level_int } = { 0 }
395        {
396           \DeclareDocumentEnvironment{__enumext_mini_page}{ m }
397             {
398                \__enumext_minipage:w [ t ] { ##1 }
399                  \legacy_if_gset_false:n { @minipage }
400                  \skip_vertical:N \c_zero_skip
401             }
402             {
403                  \skip_vertical:N \c_zero_skip
404                \__enumext_endminipage:
405             }
406        }
407   }
```

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

12.8   Compatibility with hyperref and footnotehyper

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

```
408 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
409 \hook_gset_rule:nnnn { begindocument } { enumext } { after } { hyperref }
```

__enumext_after_hyperref:
__enumext_hypertarget:nn
__enumext_phantomsection:

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

```
410 \cs_new_protected:Nn \__enumext_after_hyperref:
411   {
412      \IfPackageLoadedTF { hyperref }
413        {
414           \msg_info:nnn { enumext } { package-load } { hyperref }
415           \bool_set_true:N \l__enumext_hyperref_bool
416           \IfHyperBoolean{hyperfootnotes}
417             {
```

```
418          \typeout{hyperfootnotes=true}
419          \bool_set_true:N \l__enumext_footnotes_key_bool
420        }
421        { \typeout{hyperfootnotes=false} }
422      }
423    {  }
```

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.

```
424    \bool_if:NT \l__enumext_footnotes_key_bool
425      {
426        \IfPackageLoadedTF { footnotehyper }
427          {
428            \msg_info:nnn { enumext } { package-load } { footnotehyper }
429          }
430          {
431            \typeout{No ~ footnotehyper ~ load}
432            \typeout{Load ~ and  ~ use  ~ \string\makesavenoteenv{enumext*}}
433            \bool_set_false:N \l__enumext_footnotes_key_bool
434          }
435      }
```

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.

```
436      \bool_if:NTF \l__enumext_hyperref_bool
437        {
438          \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
439          \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
440        }
441        {
442          \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
443          \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
444        }
445    }
```

(*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.

```
446  \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
447    {
448      \protected@write \@auxout { }
449        {
450          \token_to_str:N \newlabel {#1}
451            {
452              {#2}
453              \bool_if:NT \l__enumext_hyperref_bool
454                { { \thepage } {#2} {#1} }
455                { }
456            }
457        }
458      \__enumext_hypertarget:nn {#1} { }
459      \__enumext_phantomsection:
460    }
```

(*End of definition for* __enumext_newlabel:nn.)

12.9   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.

```
461  \dim_zero_new:N \itemwidth
```

12.10 Definition of counters

__enumext_define_counters:Nn
__enumext_define_counters:cn

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.

```
462 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
463   {
464     \cs_if_exist:cTF { c@ #2 }
465       { \msg_fatal:nnn { enumext } { counters }{ #2 } }
466       {
467         \tl_set:Nn #1 { #2 }
468         \newcounter { #2 }
469       }
470   }
```

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

enumXi
enumXii
enumXiii
enumXiv
enumXv
enumXvi
enumXvii
enumXviii

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.

```
471 \__enumext_define_counters:Nn \l__enumext_counter_i_tl    { enumXi     }
472 \__enumext_define_counters:Nn \l__enumext_counter_ii_tl   { enumXii    }
473 \__enumext_define_counters:Nn \l__enumext_counter_iii_tl  { enumXiii   }
474 \__enumext_define_counters:Nn \l__enumext_counter_iv_tl   { enumXiv    }
475 \__enumext_define_counters:Nn \l__enumext_counter_v_tl    { enumXv     }
476 \__enumext_define_counters:Nn \l__enumext_counter_vi_tl   { enumXvi    }
477 \__enumext_define_counters:Nn \l__enumext_counter_vii_tl  { enumXvii   }
478 \__enumext_define_counters:Nn \l__enumext_counter_viii_tl { enumXviii  }
```

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

12.11 Definition of labels

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

__enumext_register_counter_style:Nn

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

```
479 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
480   {
481     \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
482     \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
483   }
484 \__enumext_register_counter_style:Nn \arabic { 0 }
485 \__enumext_register_counter_style:Nn \Alph   { M }
486 \__enumext_register_counter_style:Nn \alph   { m }
487 \__enumext_register_counter_style:Nn \Roman  { VIII }
488 \__enumext_register_counter_style:Nn \roman  { viii }
```

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

__enumext_label_width_by_box:Nn
__enumext_label_width_by_box:cv

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

```
489 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
490   {
491     \hbox_set:Nn \l__enumext_label_width_by_box {#2}
492     \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
493   }
494 \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.

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

```
496    {
497      \tl_clear_new:N #1
498      \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
499      \tl_gset_eq:NN \g__enumext_widest_label_tl #1
500      \tl_map_inline:Nn \g__enumext_counter_styles_tl
501        {
502          \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
503          \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
504            { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
505        }
506      \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
507        { \tl_use:N \g__enumext_widest_label_tl }
508      \tl_set_eq:cN { the #2 } #1
509    }
510  \cs_generate_variant:Nn \__enumext_label_style:Nnn { cvn }
```

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

12.12   Setting keys associated with label

font
labelsep
labelwidth
wrap-label
wrap-label*

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

```
511  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
512    {
513      \keys_define:nn { enumext / #1 }
514        {
515          font        .tl_set:c   = { l__enumext_label_font_style_#2_tl },
516          font        .value_required:n = true,
517          labelsep    .dim_set:c  = { l__enumext_labelsep_#2_dim },
518          labelsep    .initial:n  = {0.3333em},
519          labelsep    .value_required:n = true,
520          labelwidth  .dim_set:c  = { l__enumext_labelwidth_#2_dim },
521          labelwidth  .value_required:n = true,
522          wrap-label  .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
523          wrap-label  .initial:n  = {##1},
524          wrap-label  .value_required:n = true,
525          wrap-label* .code:n = {
526                                  \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
527                                  \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
528                                 },
529          wrap-label* .value_required:n = true,
530        }
531    }
532  \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

🌀 In this point, the following are set `\__enumext_wrapper_label_X:n` which will be used by `\__enumext_make_label`: for the different levels of the enumext environment and is set to `\__enumext_wrapper_label_v:n` which will be used by `\__enumext_keyans_make_label:` for keyans and keyanspic environments.

align   The align key is implemented differently for *"starred"* and *"non starred"* environments.

```
533  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
534    {
535      \keys_define:nn { enumext / #1 }
536        {
537          align .choice:,
538          align / left    .code:n =
539                            {
540                              \tl_clear:c { l__enumext_label_fill_left_#2_tl  }
541                              \tl_set:cn  { l__enumext_label_fill_right_#2_tl } { \hfill }
542                            },
543          align / right   .code:n =
544                            {
545                              \tl_set:cn  { l__enumext_label_fill_left_#2_tl  } { \hfill }
546                              \tl_clear:c { l__enumext_label_fill_right_#2_tl }
547                            },
548          align / center  .code:n =
549                            {
550                              \tl_set:cn { l__enumext_label_fill_left_#2_tl  } { \hfill }
551                              \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
552                            },
```

```
553          align / unknown .code:n =
554                              \msg_error:nneee { enumext } { unknown-choice }
555                                  { align } { left, ~ right, ~  center } { \exp_not:n {##1} },
556          align .initial:n  = left,
557          align .value_required:n  = true,
558      }
559   }
560 \clist_map_inline:nn
561   {
562     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
563   }
564   { \__enumext_tmp:nn #1 }

565 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
566   {
567     \keys_define:nn { enumext / #1 }
568       {
569         align .choice:,
570         align / left     .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
571         align / right    .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
572         align / center   .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
573         align / unknown .code:n =
574                              \msg_error:nneee { enumext } { unknown-choice }
575                                  { align } { left, ~ right, ~  center } { \exp_not:n {##1} },
576         align .initial:n = left,
577         align .value_required:n = true,
578       }
579   }
580 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }
```

(*End of definition for* align.)

12.13   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.

12.13.1   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.

```
581 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
582   {
583     \keys_define:nn { enumext / #1 }
584       {
585         label .code:n    = {
586                              \__enumext_label_style:cvn { l__enumext_label_#2_tl }
587                                { l__enumext_counter_#2_tl } {##1}
588                              \dim_set_eq:cN  { l__enumext_labelwidth_#2_dim }
589                                \l__enumext_current_widest_dim
590                            },
591         label .initial:n = #3,
592         label .value_required:n = true,
593         ref   .code:n    = \__enumext_standar_ref:n {##1},
594         ref   .value_required:n = true,
595       }
596   }
597 \__enumext_tmp:nnn { level-1 } {   i } { \arabic*.}
598 \__enumext_tmp:nnn { level-2 } {  ii } { (\alph*) }
599 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
600 \__enumext_tmp:nnn { level-4 } {  iv } { \Alph*.  }
```

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

__enumext_standar_ref:n
__enumext_standar_ref:

The __enumext_standar_ref:n first we will pass the key argument to \l__enumext_ref_key_arg_tl and we will analyze its state, if it is not *empty* we will make a copy of the current counter in \l__enumext _-ref_the_count_tl and we will execute the function __enumext_regex_counter_style: which will return the modified \l__enumext_ref_key_arg_tl and we make the value of \l__enumext_ref_the_-count_tl the same as that \l__enumext_the_counter_X_tl which contains \theenumX and finally we set \l__enumext_renew_the_count_X_tl with the renewed command.

```
601 \cs_new_protected:Npn \__enumext_standar_ref:n #1
```

```
602    {
603      \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
604      \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
605        {
606          \msg_error:nnn { enumext } { key-ref-empty } { enumext }
607        }
608        {
609          \tl_set_eq:Nc
610            \l__enumext_ref_the_count_tl { l__enumext_counter_ \__enumext_level: _tl }
611          \__enumext_regex_counter_style:
612          \tl_set_eq:Nc
613            \l__enumext_ref_the_count_tl { l__enumext_the_counter_ \__enumext_level: _tl }
614          \tl_put_right:ce { l__enumext_renew_the_count_ \__enumext_level: _tl }
615            {
616              \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
617                { \exp_not:V \l__enumext_ref_key_arg_tl }
618            }
619        }
620    }
```

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

```
621  \cs_new_protected:Nn \__enumext_standar_ref:
622    {
623      \tl_if_empty:cF { l__enumext_renew_the_count_ \__enumext_level: _tl }
624        {
625          \tl_use:c { l__enumext_renew_the_count_ \__enumext_level: _tl }
626        }
627    }
```

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

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

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

<div style="text-align:right">

label
ref
\l__enumext_label_vii_tl
\l__enumext_label_viii_tl
</div>

```
628  \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
629    {
630      \keys_define:nn { enumext / #1 }
631        {
632          label .code:n     = {
633                                \__enumext_label_style:cvn { l__enumext_label_#2_tl }
634                                  { l__enumext_counter_#2_tl } {##1}
635                                \dim_set_eq:cN  { l__enumext_labelwidth_#2_dim }
636                                  \l__enumext_current_widest_dim
637                              },
638          label .initial:n = #3,
639          label .value_required:n = true,
640          ref    .code:n     = \__enumext_starred_ref:n {##1},
641          ref    .value_required:n = true,
642        }
643    }
644  \__enumext_tmp:nnn { enumext* } {  vii } { \arabic*.}
645  \__enumext_tmp:nnn { keyans*  } { viii } { \Alph*) }
```

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

<div style="text-align:right">

__enumext_starred_ref:n
__enumext_starred_ref:
</div>

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

```
646  \cs_new_protected:Npn \__enumext_starred_ref:n #1
647    {
648      \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
649      \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
650        {
651          \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
652            {
653              \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
654            }
655            {
656              \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
657              \__enumext_regex_counter_style:
658              \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
659              \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
```

```
660                   {
661                       \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
662                         { \exp_not:V \l__enumext_ref_key_arg_tl }
663                   }
664               }
665           }
666       \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
667           {
668             \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
669               {
670                 \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
671               }
672               {
673                 \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
674                 \__enumext_regex_counter_style:
675                 \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
676                 \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
677                   {
678                       \exp_not:N \renewcommand  { \exp_not:V \l__enumext_ref_the_count_tl }
679                         { \exp_not:V \l__enumext_ref_key_arg_tl }
680                   }
681               }
682           }
683       }
```

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.

```
684   \cs_new_protected:Nn \__enumext_starred_ref:
685       {
686         \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
687           {
688             \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
689               {
690                 \tl_use:N \l__enumext_renew_the_count_vii_tl
691               }
692           }
693         \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
694           {
695             \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
696               {
697                 \tl_use:N \l__enumext_renew_the_count_viii_tl
698               }
699           }
700       }
```

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

12.13.3 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 and ref key. The keyanspic environment use the same ⟨*label*⟩ as the keyans environment.

```
701   \keys_define:nn { enumext / keyans }
702       {
703         label .code:n     = {
704                               \__enumext_label_style:cvn { l__enumext_label_v_tl }
705                                 { l__enumext_counter_v_tl } {#1}
706                               \dim_set_eq:cN  { l__enumext_labelwidth_v_dim }
707                                 \l__enumext_current_widest_dim
708                               \__enumext_label_style:cvn { l__enumext_label_vi_tl }
709                                 { l__enumext_counter_vi_tl } {#1}
710                               \dim_set_eq:cN  { l__enumext_labelwidth_v_dim }
711                                 \l__enumext_current_widest_dim
712                             },
713         label .initial:n = \Alph*),
714         label .value_required:n = true,
715         ref   .code:n    = \__enumext_keyans_ref:n {#1},
716         ref   .value_required:n = true,
717       }
```

(*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.

```
718  \cs_new_protected:Npn \__enumext_keyans_ref:n #1
719    {
720      \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
721      \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
722        {
723          \msg_error:nnn { enumext } { key-ref-empty } { keyans }
724        }
725        {
726          \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
727          \__enumext_regex_counter_style:
728          \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
729          \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
730            {
731              \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
732                { \exp_not:V \l__enumext_ref_key_arg_tl }
733            }
734        }
735    }
```

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

```
736  \cs_new_protected:Nn \__enumext_keyans_ref:
737    {
738      \tl_if_empty:NF \l__enumext_renew_the_count_v_tl
739        {
740          \tl_use:N \l__enumext_renew_the_count_v_tl
741        }
742    }
```

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

12.14 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.

```
743  \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
744    {
745      \__enumext_if_is_int:nTF { #3 }
746        {
747          \int_set:Nn #2 {#3}
748        }
749        {
750          \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
751            { \int_set:Nn #2 { \int_from_alph:n {#3} } }
752          \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
753            { \int_set:Nn #2  { \int_from_roman:n {#3} } }
754        }
755    }
756  \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.

```
757  \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
758    {
759      \__enumext_if_is_int:nTF {#4}
760        {
761          \setcounter{enumX#1} { #4 }
```

```
762        }
763        {
764            \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
765                { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
766            \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
767                { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
768        }
769        \__enumext_label_width_by_box:cv
770            { l__enumext_labelwidth_#1_dim } { l__enumext_label_#1_tl }
771    }
772  \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.

```
773  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
774    {
775      \keys_define:nn { enumext / #1 }
776        {
777          start* .code:n    = {
778                               \__enumext_start_from:ccn
779                                 { l__enumext_label_#2_tl }
780                                 { l__enumext_start_#2_int } {##1}
781                             },
782          start* .value_required:n = true,
783          start  .code:n    = {
784                               \__enumext_start_from:cce
785                                 { l__enumext_label_#2_tl }
786                                 { l__enumext_start_#2_int } { \int_eval:n {##1} }
787                             },
788          start  .initial:n = 1,
789          start  .value_required:n = true,
790          widest .code:n    = {
791                               \__enumext_widest_from:nccn {#2}
792                                 { l__enumext_label_#2_tl }
793                                 { l__enumext_labelwidth_#2_dim } {##1}
794                             },
795          widest .value_required:n = true,
796        }
797    }
798  \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

12.15   Setting keys for vertical spaces

topsep
partopsep
parsep
noitemsep
nosep

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

```
799  \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
800    {
801      \keys_define:nn { enumext / #1 }
802        {
803          topsep    .skip_set:c = { l__enumext_topsep_#2_skip },
804          topsep    .initial:n  = {#3},
805          topsep    .value_required:n = true,
806          partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
807          partopsep .initial:n  = {#4},
808          partopsep .value_required:n = true,
809          parsep    .skip_set:c = { l__enumext_parsep_#2_skip },
810          parsep    .initial:n  = {#5},
811          parsep    .value_required:n = true,
812          itemsep   .skip_set:c = { l__enumext_itemsep_#2_skip },
813          itemsep   .initial:n  = {#6},
814          itemsep   .value_required:n = true,
815          noitemsep .meta:n     = { itemsep = 0pt, parsep = 0pt },
816          noitemsep .value_forbidden:n = true,
817          nosep     .meta:n     = {
818                                   itemsep = 0pt, parsep= 0pt,
819                                   topsep = 0pt, partopsep = 0pt,
820                                 },
```

```
821          nosep      .value_forbidden:n = true,
822        }
823    }
```

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

```
824  \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
825    { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
826    { 4.0pt plus 2.0pt minus 1.0pt }
827  \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
828    { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
829    { 2.0pt plus 1.0pt minus 1.0pt }
830  \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
831    { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
832  \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
833    { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
834  \__enumext_tmp:nnnnnn { keyans  } { v }{ 4.0pt plus 2.0pt minus 1.0pt }
835    { 2.0pt plus 1.0pt minus 1.0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
836    { 2.0pt plus 1.0pt minus 1.0pt }
837  \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
838    { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
839    { 4.0pt plus 2.0pt minus 1.0pt }
840  \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
841    { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
842    { 2.0pt plus 1.0pt minus 1.0pt }
```

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

12.16   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:` and then apply `\vspace{-\baselineskip}` and set `topsep=0pt` for the *"first level"* of the nested `enumext` or `enumext*` environments.

base-fix
__enumext_nested_base_line_fix:

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

```
843  \cs_set_protected:Npn \__enumext_tmp:n #1
844    {
845      \keys_define:nn { enumext / #1 }
846        {
847          base-fix .bool_set:N = \l__enumext_base_line_fix_bool,
848          base-fix .initial:n  = false,
849          base-fix .value_forbidden:n = true,
850        }
851    }
852  \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

The function `\__enumext_nested_base_line_fix:` will be in charge of applying the baseline correction and adjusting the ⟨*keys*⟩. This function is passed to the function `\__enumext_parse_keys:n` in the `enumext` environment definition (§12.38) and to the function `\__enumext_parse_keys_vii:n` in the `enumext*` environment definition (§12.43).

```
853  \cs_new_protected:Nn \__enumext_nested_base_line_fix:
854    {
855      \bool_lazy_and:nnT
856        { \bool_if_p:N \l__enumext_standar_first_bool }
857        { \bool_if_p:N \l__enumext_base_line_fix_bool }
858        {
859          \mode_leave_vertical:
860          \vspace { -\baselineskip }
861          \keys_set:nn { enumext / level-1 }
862            {
863              topsep = 0pt, above = 0pt, above* = 0pt,
864            }
865        }
866      \bool_lazy_and:nnT
867        { \bool_if_p:N \l__enumext_starred_first_bool }
868        { \bool_if_p:N \l__enumext_base_line_fix_bool }
869        {
870          \mode_leave_vertical:
871          \vspace { -\baselineskip }
872          \keys_set:nn { enumext / enumext* }
873            {
874              topsep = 0pt, above = 0pt, above* = 0pt,
```

```
875              }
876          }
877          \bool_set_false:N \l__enumext_base_line_fix_bool
878      }
```

💣 This key is enabled by default in the command `\printkeyans` (§12.46).

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

12.17   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.

```
879 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
880   {
881     \keys_define:nn { enumext / #1 }
882       {
883         itemindent    .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
884         itemindent    .value_required:n = true,
885         rightmargin   .dim_set:c = { l__enumext_rightmargin_#2_dim },
886         rightmargin   .value_required:n = true,
887         listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
888         listparindent .value_required:n = true,
889         list-offset   .dim_set:c = { l__enumext_listoffset_#2_dim },
890         list-offset   .value_required:n = true,
891         list-indent   .code:n    =
892                           \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
893                           \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
894         list-indent   .value_required:n = true,
895       }
896   }
897 \clist_map_inline:nn
898   {
899     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
900   }
901   { \__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.

```
902 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
903   {
904     \keys_define:nn { enumext / #1 }
905       {
906         itemindent    .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
907         itemindent    .value_required:n = true,
908         rightmargin   .dim_set:c = { l__enumext_rightmargin_#2_dim },
909         rightmargin   .value_required:n = true,
910         listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
911         listparindent .value_required:n = true,
912         list-offset   .dim_set:c = { l__enumext_listoffset_#2_dim },
913         list-offset   .value_required:n = true,
914         list-indent   .meta:n    = { list-offset = ##1 },
915         list-indent   .value_required:n = true,
916       }
917   }
918 \clist_map_inline:nn
919   {
920     {enumext*}{vii}, {keyans*}{viii}
921   }
922   { \__enumext_tmp:nn #1 }
```

12.17.1   Functions for setting the fake `itemindent`

__enumext_fake_item:
__enumext_keyans_fake_item:
__enumext_fake_item_vii:
__enumext_fake_item_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.

```
923 \cs_set_protected:Nn \__enumext_fake_item:
924   {
925     \dim_compare:nNnT
926       { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
927       >
```

```
928       { \c_zero_dim }
929       {
930         \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
931           {
932             \exp_not:N \mode_leave_vertical:
933             \exp_not:n { \skip_horizontal:n }
934               { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
935             \ignorespaces
936           }
937       }
938     }
939 \cs_set_protected:Nn \__enumext_keyans_fake_item:
940   {
941     \dim_compare:nNnT
942       { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
943       {
944         \tl_set:Ne \l__enumext_fake_item_indent_v_tl
945           {
946             \exp_not:N \mode_leave_vertical:
947             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
948           }
949       }
950   }
951 \cs_set_protected:Nn \__enumext_fake_item_vii:
952   {
953     \dim_compare:nNnT
954       { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
955       {
956         \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
957           {
958             \exp_not:N \mode_leave_vertical:
959             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
960           }
961       }
962   }
963 \cs_set_protected:Nn \__enumext_fake_item_viii:
964   {
965     \dim_compare:nNnT
966       { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
967       {
968         \tl_set:Ne \l__enumext_fake_item_indent_viii_tl
969           {
970             \exp_not:N \mode_leave_vertical:
971             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
972           }
973       }
974   }
```

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

12.18   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.

```
975 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
976   {
977     \keys_define:nn { enumext / #1 }
978       {
979         show-length .bool_set:c = { l__enumext_show_length_#2_bool },
980         show-length .initial:n  = false,
981       }
982   }
983 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

12.19   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

```
984  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
985    {
986      \keys_define:nn { enumext / #1 }
987        {
988          before   .tl_set:c   = { l__enumext_before_no_starred_key_#2_tl },
989          before   .value_required:n = true,
990          before*  .tl_set:c   = { l__enumext_before_starred_key_#2_tl },
991          before*  .value_required:n = true,
992          after    .tl_set:c   = { l__enumext_after_stop_list_#2_tl },
993          after    .value_required:n = true,
994          first    .tl_set:c   = { l__enumext_after_list_args_#2_tl },
995          first    .value_required:n = true,
996        }
997    }
998  \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

12.19.1 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*⟩}.

```
999  \cs_new_protected:Nn \__enumext_before_args_exec:
1000   {
1001     \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
1002   }
```

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*⟩}}

```
1003  \cs_new_protected:Nn \__enumext_before_keys_exec:
1004   {
1005     \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
1006   }
```

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

```
1007  \cs_new_protected:Nn \__enumext_after_stop_list:
1008   {
1009     \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
1010   }
```

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.

```
1011  \cs_new_protected:Nn \__enumext_after_args_exec:
1012   {
1013     \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
1014   }
```

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

12.19.2 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.

```
1015  \cs_new_protected:Nn \__enumext_before_args_exec_v:
1016   {
1017     \tl_use:N \l__enumext_before_starred_key_v_tl
1018   }
1019  \cs_new_protected:Nn \__enumext_before_keys_exec_v:
1020   {
1021     \tl_use:N \l__enumext_before_no_starred_key_v_tl
1022   }
1023  \cs_new_protected:Nn \__enumext_after_stop_list_v:
1024   {
1025     \tl_use:N \l__enumext_after_stop_list_v_tl
1026   }
1027  \cs_new_protected:Nn \__enumext_after_args_exec_v:
1028   {
1029     \tl_use:N \l__enumext_after_list_args_v_tl
1030   }
```

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

12.19.3 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.

```
1031  \cs_new_protected:Nn \__enumext_before_args_exec_vii:
1032    {
1033      \tl_use:N \l__enumext_before_starred_key_vii_tl
1034    }
1035  \cs_new_protected:Nn \__enumext_before_args_exec_viii:
1036    {
1037      \tl_use:N \l__enumext_before_starred_key_viii_tl
1038    }
1039  \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
1040    {
1041      \tl_use:N \l__enumext_before_no_starred_key_vii_tl
1042    }
1043  \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
1044    {
1045      \tl_use:N \l__enumext_before_no_starred_key_viii_tl
1046    }
1047  \cs_new_protected:Nn \__enumext_after_stop_list_vii:
1048    {
1049      \tl_use:N \l__enumext_after_stop_list_vii_tl
1050    }
1051  \cs_new_protected:Nn \__enumext_after_stop_list_viii:
1052    {
1053      \tl_use:N \l__enumext_after_stop_list_viii_tl
1054    }
1055  \cs_new_protected:Nn \__enumext_after_args_exec_vii:
1056    {
1057      \tl_use:N \l__enumext_after_list_args_vii_tl
1058    }
1059  \cs_new_protected:Nn \__enumext_after_args_exec_viii:
1060    {
1061      \tl_use:N \l__enumext_after_list_args_viii_tl
1062    }
```

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

12.20 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.

```
1063  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1064    {
1065      \keys_define:nn { enumext / #1 }
1066        {
1067          mini-env    .dim_set:c  = { l__enumext_minipage_right_#2_dim },
1068          mini-env    .value_required:n = true,
1069          mini-sep    .dim_set:c  = { l__enumext_minipage_hsep_#2_dim },
1070          mini-sep    .initial:n  = 0.3333em,
1071          mini-sep    .value_required:n = true,
1072          columns-sep .dim_set:c  = { l__enumext_columns_sep_#2_dim },
1073          columns-sep .value_required:n = true,
1074          columns     .int_set:c  = { l__enumext_columns_#2_int },
1075          columns     .initial:n  = 1,
1076          columns     .value_required:n = true,
1077        }
1078    }
1079  \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.

```
1080  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1081    {
1082      \keys_define:nn { enumext / #1 }
1083        {
1084          mini-right  .tl_gset:c = { g__enumext_miniright_code_#2_tl },
1085          mini-right  .value_required:n = true,
1086          mini-right* .code:n    = {
```

```
1087                                    \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
1088                                    \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1089                                },
1090            mini-right* .value_required:n = true,
1091        }
1092    }
1093 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }
```

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

12.21 Adjustment of vertical spaces for multicols

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

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

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

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

12.21.1 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*.

```
1094 \cs_new_protected:Nn \__enumext_multi_set_vskip:
1095    {
1096       \skip_set:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1097          {
1098             \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1099          }
1100       \skip_set:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1101          {
1102             \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1103          }
1104       \__enumext_add_pre_parsep:
1105    }
```

(*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*.

```
1106 \cs_new_protected:Nn \__enumext_add_pre_parsep:
1107    {
1108       \int_case:nn { \l__enumext_level_int }
1109          {
1110             { 2 }{
1111                \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1112                   {
1113                      \skip_add:Nn \l__enumext_multicols_above_ii_skip { \l__enumext_parsep_i_skip }
1114                   }
1115                }
1116             { 3 }{
1117                \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1118                   {
```

```
1119                        \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_ii_skip
1120                    }
1121                }
1122            { 4 }{
1123                \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1124                {
1125                    \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip
1126                }
1127            }
1128        }
1129    }
```

(*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*⟩.

```
1130  \cs_new_protected:Nn \__enumext_multi_addvspace:
1131    {
1132      \__enumext_multi_set_vskip:
1133      \mode_if_vertical:T
1134        {
1135          \skip_add:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1136            {
1137              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1138            }
1139          \skip_add:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1140            {
1141              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1142            }
1143        }
1144      %%\__enumext_unskip_unkern:
1145      \par\nopagebreak
1146      \addvspace{ \skip_use:c { l__enumext_multicols_above_ \__enumext_level: _skip } }
1147    }
```

(*End of definition for* __enumext_multi_addvspace:.)

12.21.2   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.

```
1148  \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1149    {
1150      \skip_set:Nn \l__enumext_multicols_above_v_skip
1151        {
1152          \l__enumext_topsep_v_skip
1153        }
1154      \skip_set:Nn \l__enumext_multicols_below_v_skip
1155        {
1156          \l__enumext_topsep_v_skip
1157        }
1158    }
1159  \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1160    {
1161      \__enumext_keyans_multi_set_vskip:
1162      \mode_if_vertical:T
1163        {
1164          \skip_add:Nn \l__enumext_multicols_above_v_skip
1165            {
1166              \skip_use:N \l__enumext_partopsep_v_skip
1167            }
1168          \skip_add:Nn \l__enumext_multicols_below_v_skip
1169            {
1170              \skip_use:N \l__enumext_partopsep_v_skip
1171            }
1172        }
1173      \__enumext_unskip_unkern:
1174      \par\nopagebreak
1175      \addvspace{ \l__enumext_multicols_above_v_skip }
1176    }
```

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

12.22 Adjustment of vertical spaces for minipage

When nesting a *"list environment"* within the minipage environment, the values of the *"vertical spaces"* are lost. Graphically it can be seen like in the figure 8.
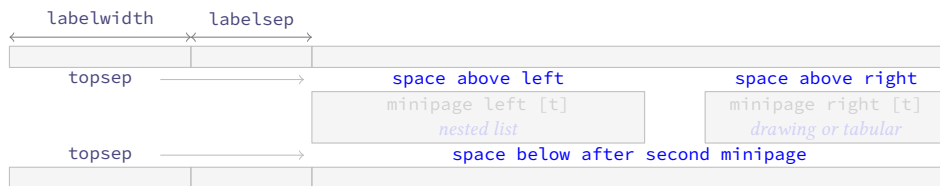
Figure 8: Representation of the minipage spacing adjustment for a nested level.

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

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

💣 Again I must make clear that the implementation here is a *"bit questionable"*, but hunting the spaces (glue) produced by the minipage environment is quite complicated, even more if multicols it is nested. The setting of the values was more *"trial and error"* (aprox to \strutbox), using the help of the lua-visual-debug[14] package, again my attempts to find the correct values using \showoutput and \showboxdepth absolutely failed.

12.22.1 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.

```
1177  \cs_new_protected:Nn \__enumext_minipage_set_skip:
1178    {
1179      \skip_set:Nn \l__enumext_minipage_right_skip
1180        {
1181          \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1182        }
1183      \mode_if_vertical:T
1184        {
1185          \skip_add:Nn \l__enumext_minipage_right_skip
1186            {
1187              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1188            }
1189        }
1190      \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:.

```
1191      \skip_set_eq:cN
1192        { l__enumext_multicols_above_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1193      \skip_set_eq:cN
1194        { l__enumext_multicols_below_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1195      \__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.

```
1196      \int_compare:nNnT
1197        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1198        {
1199          \skip_zero:N \topskip
1200          \skip_set_eq:Nc \multicolsep { l__enumext_multicols_above_ \__enumext_level: _skip }
1201        }
1202    }
```

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*.

```
1203  \cs_new_protected:Nn \__enumext_minipage_add_space:
1204    {
1205      \__enumext_minipage_set_skip:
1206      \__enumext_unskip_unkern:
1207      \mode_if_vertical:TF
1208        {
1209          \nopagebreak\nointerlineskip
1210        }
1211        {
1212          \par\nopagebreak\nointerlineskip
1213          \skip_zero:c { l__enumext_partopsep_ \__enumext_level: _skip }
1214        }
1215      \int_compare:nNnTF
1216        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1217        {
1218          \addvspace{ 0.445\box_ht:N \strutbox }
1219        }
1220        {
1221          \addvspace{ 0.250\box_ht:N \strutbox }
1222        }
1223    }
```

(*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.

```
1224  \cs_new_protected:Nn \__enumext_pre_itemsep_skip:
1225    {
1226      \int_case:nn { \l__enumext_level_int }
1227        {
1228          { 2 }{
1229            \skip_if_eq:nnTF
1230              { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1231              {
1232                \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1233                \skip_set:Nn \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1234              }
1235              {
1236                \dim_compare:nNnT
1237                  { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1238                  {
1239                    \skip_sub:Nn
1240                      \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1241                    \skip_sub:Nn
1242                      \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1243                    \skip_add:Nn
1244                      \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1245                    \skip_add:Nn
1246                      \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1247                  }
1248                \dim_compare:nNnT
1249                  { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1250                  {
1251                    \skip_set:Nn \l_tmpa_skip
1252                      {
1253                        \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1254                      }
1255                    \skip_sub:Nn
1256                      \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1257                    \skip_sub:Nn
1258                      \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1259                    \skip_add:Nn
1260                      \l__enumext_minipage_after_skip
1261                      { 0.150\box_ht:N \strutbox + \l_tmpa_skip }
1262                    \skip_add:Nn
1263                      \l__enumext_multicols_below_ii_skip
1264                      { 0.350\box_ht:N \strutbox + \l_tmpa_skip }
1265                  }
```

```
1266                              }
1267                    }
1268           { 3 }{
1269              \skip_if_eq:nnTF
1270                 { \l__enumext_itemsep_ii_skip } { \c_zero_skip }
1271                 {
1272                    \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1273                    \skip_set:Nn \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1274                 }
1275                 {
1276                    \dim_compare:nNnT
1277                      { \l__enumext_itemsep_ii_skip } < { \l__enumext_minipage_after_skip }
1278                      {
1279                         \skip_sub:Nn
1280                            \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1281                         \skip_sub:Nn
1282                            \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1283                         \skip_add:Nn
1284                            \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1285                         \skip_add:Nn
1286                            \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1287                      }
1288                    \dim_compare:nNnT
1289                      { \l__enumext_itemsep_ii_skip } > { \l__enumext_minipage_after_skip }
1290                      {
1291                         \skip_set:Nn \l_tmpa_skip
1292                           {
1293                              \l__enumext_itemsep_ii_skip - \l__enumext_minipage_after_skip
1294                           }
1295                         \skip_sub:Nn
1296                            \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1297                         \skip_sub:Nn
1298                            \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1299                         \skip_add:Nn
1300                            \l__enumext_minipage_after_skip
1301                            { 0.150\box_ht:N \strutbox + \l_tmpa_skip }
1302                         \skip_add:Nn
1303                            \l__enumext_multicols_below_iii_skip
1304                            { 0.350\box_ht:N \strutbox + \l_tmpa_skip }
1305                      }
1306                 }
1307              }
1308           { 4 }{
1309              \skip_if_eq:nnTF { \l__enumext_itemsep_iii_skip } { \c_zero_skip }
1310                 {
1311                    \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1312                    \skip_set:Nn \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1313                 }
1314                 {
1315                    \dim_compare:nNnT
1316                      { \l__enumext_itemsep_iii_skip } < { \l__enumext_minipage_after_skip }
1317                      {
1318                         \skip_sub:Nn
1319                            \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1320                         \skip_sub:Nn
1321                            \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1322                         \skip_add:Nn
1323                            \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1324                         \skip_add:Nn
1325                            \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1326                      }
1327                    \dim_compare:nNnT
1328                      { \l__enumext_itemsep_iii_skip } > { \l__enumext_minipage_after_skip }
1329                      {
1330                         \skip_set:Nn \l_tmpa_skip
1331                           {
1332                              \l__enumext_itemsep_iii_skip - \l__enumext_minipage_after_skip
1333                           }
1334                         \skip_sub:Nn
1335                            \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1336                         \skip_sub:Nn
```

```
1337                              \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1338                          \skip_add:Nn
1339                            \l__enumext_minipage_after_skip
1340                            { 0.150\box_ht:N \strutbox + \l_tmpa_skip }
1341                          \skip_add:Nn
1342                            \l__enumext_multicols_below_iv_skip
1343                            { 0.350\box_ht:N \strutbox + \l_tmpa_skip }
1344                        }
1345                      }
1346                    }
1347                  }
1348                }
```

(*End of definition for* __enumext_pre_itemsep_skip:.)

12.22.2   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.

```
1349  \cs_new_protected:Nn \__enumext_keyans_minipage_set_skip:
1350    {
1351      \skip_zero:N \l__enumext_minipage_after_skip
1352      \skip_zero:N \l__enumext_minipage_left_skip
1353      \skip_zero:N \l__enumext_minipage_right_skip
1354      \skip_set:Nn \l__enumext_minipage_right_skip
1355        {
1356          \l__enumext_topsep_v_skip
1357        }
1358      \mode_if_vertical:T
1359        {
1360          \skip_add:Nn \l__enumext_minipage_right_skip
1361            {
1362              \l__enumext_partopsep_v_skip
1363            }
1364        }
1365      \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip
1366      \skip_set_eq:NN \l__enumext_multicols_above_v_skip \l__enumext_minipage_right_skip
1367      \skip_set_eq:NN \l__enumext_multicols_below_v_skip \l__enumext_minipage_right_skip
1368      \__enumext_keyans_pre_itemsep_skip:
1369      \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
1370        {
1371          \skip_zero:N \topskip
1372          \skip_set_eq:NN \multicolsep \l__enumext_minipage_right_skip
1373        }
1374    }
1375  \cs_new_protected:Nn \__enumext_keyans_minipage_add_space:
1376    {
1377      \__enumext_keyans_minipage_set_skip:
1378      \__enumext_unskip_unkern:
1379      \mode_if_vertical:TF
1380        {
1381          \nopagebreak\nointerlineskip
1382        }
1383        {
1384          \par\nopagebreak\nointerlineskip
1385          \skip_zero:N \l__enumext_partopsep_v_skip
1386        }
1387      \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1388        {
1389          \addvspace{ 0.445\box_ht:N \strutbox }
1390        }
1391        {
1392          \addvspace{ 0.250\box_ht:N \strutbox }
1393        }
1394    }
1395  \cs_new_protected:Nn \__enumext_keyans_pre_itemsep_skip:
1396    {
1397      \skip_if_eq:nnTF
1398        { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1399        {
1400          \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
```

```
1401        \skip_set:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1402      }
1403      {
1404        \dim_compare:nNnT
1405          { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1406          {
1407            \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1408            \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1409            \skip_add:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1410            \skip_add:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1411          }
1412        \dim_compare:nNnT
1413          { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1414          {
1415            \skip_set:Nn \l_tmpa_skip
1416              {
1417                \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1418              }
1419            \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1420            \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1421            \skip_add:Nn \l__enumext_minipage_after_skip
1422              { 0.150\box_ht:N \strutbox + \l_tmpa_skip }
1423            \skip_add:Nn \l__enumext_multicols_below_v_skip
1424              { 0.350\box_ht:N \strutbox + \l_tmpa_skip }
1425          }
1426      }
1427    }
```

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

12.22.3   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*.

```
1428 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1429    {
1430      \skip_zero_new:N \l__enumext_minipage_left_skip
1431      \skip_gzero_new:N \g__enumext_minipage_right_skip
1432      \skip_gzero_new:N \g__enumext_minipage_after_skip
1433      \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1434        {
1435          \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1436          \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1437        }
1438        {
1439          \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1440          \skip_gset:Nn \g__enumext_minipage_right_skip
1441            {
1442              \l__enumext_topsep_vii_skip
1443            }
1444          \skip_gset:Nn \g__enumext_minipage_after_skip
1445            {
1446              0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1447            }
1448        }
1449    }
1450 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1451    {
1452      \skip_zero_new:N \l__enumext_minipage_after_skip
1453      \skip_zero_new:N \l__enumext_minipage_left_skip
1454      \skip_zero_new:N \l__enumext_minipage_right_skip
1455      \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1456        {
1457          \skip_set:Nn \l__enumext_minipage_left_skip
1458            {
1459              0.5\box_dp:N \strutbox
1460            }
1461          \skip_set:Nn \l__enumext_minipage_right_skip
1462            {
1463              \l__enumext_partopsep_viii_skip
```

```
1464              }
1465          \skip_set:Nn \l__enumext_minipage_after_skip
1466            {
1467               1.6\box_dp:N \strutbox
1468            }
1469        }
1470        {
1471          \skip_set:Nn \l__enumext_minipage_left_skip
1472            {
1473               0.5875\box_dp:N \strutbox
1474            }
1475          \skip_set:Nn \l__enumext_minipage_right_skip
1476            {
1477               \l__enumext_topsep_viii_skip
1478            }
1479          \skip_set:Nn \l__enumext_minipage_after_skip
1480            {
1481               0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1482            }
1483        }
1484    }
```

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

__enumext_mini_addvspace_vii:
__enumext_mini_addvspace_viii:

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

```
1485 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1486    {
1487      \__enumext_mini_set_vskip_vii:
1488      \par\nopagebreak
1489      \addvspace { \l__enumext_minipage_left_skip }
1490    }
1491 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1492    {
1493      \__enumext_mini_set_vskip_viii:
1494      \par\nopagebreak
1495      \addvspace { \l__enumext_minipage_left_skip }
1496    }
```

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

12.22.4 The command \miniright

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

\miniright

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

```
1497 \NewDocumentCommand \miniright { s }
1498    {
1499      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1500        {
1501          \msg_error:nnn { enumext } { wrong-miniright-place }
1502        }
1503    % outside
1504      \bool_lazy_and:nnT
1505        { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
1506        { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
1507        {
1508          \msg_error:nnn { enumext } { wrong-miniright-place }
1509        }
1510    % starred env
1511      \bool_if:NT \l__enumext_starred_bool
1512        {
```

```
1513        \msg_error:nnn { enumext } { wrong-miniright-starred }
1514      }
1515    \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
1516      {
1517        \__enumext_keyans_mini_right_cmd:n {#1}
1518      }
1519    { \__enumext_mini_right_cmd:n {#1} }
1520  }
```

(*End of definition for* \miniright. *This function is documented on page 10.*)

__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.

```
1521  \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1522    {
1523      \dim_compare:nNnTF
1524        { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1525        {
1526          \__enumext_multicols_stop:
1527          \int_compare:nNnT
1528            { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } = { 1 }
1529            {
1530              \par\addvspace{ \l__enumext_minipage_after_skip }
1531            }
1532          \end__enumext_mini_page
1533          \hfill
1534          \__enumext_mini_page{ \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } }
1535            \par\nointerlineskip
1536            \addvspace { \l__enumext_minipage_right_skip }
1537            \bool_if:nF {#1}
1538              {
1539                \centering
1540              }
1541            \int_gzero:N \g__enumext_minipage_stat_int
1542        }
1543        { \msg_error:nnn { enumext } { wrong-miniright-use } }
1544      % paranoia
1545      \RenewDocumentCommand \miniright { s }
1546        {
1547          \msg_error:nn { enumext } { many-miniright-used }
1548        }
1549    }
```

(*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.

```
1550  \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1551    {
1552      \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1553        {
1554          \__enumext_keyans_multicols_stop:
1555          \int_compare:nNnT { \l__enumext_columns_v_int } = { 1 }
1556            {
1557              \par\addvspace{ \l__enumext_minipage_after_skip }
1558            }
1559          \end__enumext_mini_page
1560          \hfill
1561          \__enumext_mini_page{ \l__enumext_minipage_right_v_dim }
1562            \par\nointerlineskip
1563            \addvspace { \l__enumext_minipage_right_skip }
1564            \bool_if:nF {#1}
1565              {
1566                \centering
1567              }
1568            \int_gzero:N \g__enumext_minipage_stat_int
```

```
1569        }
1570      { \msg_error:nnn { enumext } { wrong-miniright-use } }
1571    % paranoia
1572    \RenewDocumentCommand \miniright { s }
1573      {
1574        \msg_error:nn { enumext } { many-miniright-used }
1575      }
1576  }
```

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

12.23 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.

```
1577 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1578   {
1579     \keys_define:nn { enumext / #1 }
1580       {
1581         above  .skip_set:c = { l__enumext_vspace_above_#2_skip },
1582         above  .value_required:n = true,
1583         above* .code:n     = \bool_set_true:c { l__enumext_vspace_a_star_#2_bool }
1584                               \keys_set:nn { enumext / #1 } { above = {##1} },
1585         above* .value_required:n = true,
1586         below  .skip_set:c = { l__enumext_vspace_below_#2_skip },
1587         below  .value_required:n = true,
1588         below* .code:n     = \bool_set_true:c { l__enumext_vspace_b_star_#2_bool }
1589                               \keys_set:nn { enumext / #1 } { below = {##1} },
1590         below* .value_required:n = true,
1591       }
1592   }
1593 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

12.23.1 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.

```
1594 \cs_new_protected:Nn \__enumext_vspace_above:
1595   {
1596     \skip_if_eq:nnF
1597       { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1598       {
1599         \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
1600           {
1601             \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1602           }
1603           {
1604             \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1605           }
1606       }
1607   }
```

(*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.

```
1608 \cs_new_protected:Nn \__enumext_vspace_below:
1609   {
1610     \skip_if_eq:nnF
1611       { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1612       {
1613         \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1614           {
1615             \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1616           }
```

```
1617                    {
1618                        \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1619                    }
1620                }
1621            }
```

(*End of definition for* __enumext_vspace_below:.)

12.23.2   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.

```
1622  \cs_new_protected:Nn \__enumext_vspace_above_v:
1623    {
1624       \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1625         {
1626            \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1627              {
1628                 \vspace*{ \l__enumext_vspace_above_v_skip }
1629              }
1630              { \vspace { \l__enumext_vspace_above_v_skip } }
1631         }
1632    }
```

(*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.

```
1633  \cs_new_protected:Nn \__enumext_vspace_below_v:
1634    {
1635       \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1636         {
1637            \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1638              {
1639                 \vspace*{ \l__enumext_vspace_below_v_skip }
1640              }
1641              { \vspace { \l__enumext_vspace_below_v_skip } }
1642         }
1643    }
```

(*End of definition for* __enumext_vspace_below_v:.)

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

__enumext_vspace_above_vii: The functions __enumext_vspace_above_vii: and __enumext_vspace_above_viii: apply the *verti-*
__enumext_vspace_above_viii: *cal space above* the enumext* and keyans* environments set by the above and above* keys.

```
1644  \cs_new_protected:Nn \__enumext_vspace_above_vii:
1645    {
1646       \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1647         {
1648            \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1649              {
1650                 \vspace*{ \l__enumext_vspace_above_vii_skip }
1651              }
1652              { \vspace { \l__enumext_vspace_above_vii_skip } }
1653         }
1654    }
1655  \cs_new_protected:Nn \__enumext_vspace_above_viii:
1656    {
1657       \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1658         {
1659            \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1660              {
1661                 \vspace*{ \l__enumext_vspace_above_viii_skip }
1662              }
1663              { \vspace { \l__enumext_vspace_above_viii_skip } }
1664         }
1665    }
```

(*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.

```
1666 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1667   {
1668     \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1669       {
1670         \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1671           {
1672             \vspace*{ \l__enumext_vspace_below_vii_skip }
1673           }
1674           { \vspace { \l__enumext_vspace_below_vii_skip } }
1675       }
1676   }
1677 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1678   {
1679     \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1680       {
1681         \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1682           {
1683             \vspace*{ \l__enumext_vspace_below_viii_skip }
1684           }
1685           { \vspace { \l__enumext_vspace_below_viii_skip } }
1686       }
1687   }
```

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

12.24 Setting series, resume and resume* keys

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

series
resume
resume*

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

```
1688 \cs_set_protected:Npn \__enumext_tmp:n #1
1689   {
1690     \keys_define:nn { enumext / #1 }
1691       {
1692         series  .str_set:N = \l__enumext_series_str,
1693         series  .value_required:n = true,
1694         resume  .code:n = \__enumext_resume_series:n {##1},
1695         resume* .code:n = \__enumext_resume_starred:,
1696         resume* .value_forbidden:n = true,
1697       }
1698   }
1699 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

12.24.1 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 value passed to the environment.

```
1700 \cs_new:Npn \__enumext_filter_series:n #1
1701   {
1702     \use:e
1703       {
1704         \keyval_parse:NNn
1705           \__enumext_filter_series_key:n
1706           \__enumext_filter_series_pair:nn {#1}
1707       }
1708   }
```

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

```
1709 \cs_new:Npn \__enumext_filter_series_key:n #1
1710   {
1711     \str_case:nnF {#1}
1712       {
1713         { resume } {} { resume* } {} { base-fix } {}
1714       }
1715       { , { \exp_not:n {#1} } }
1716   }
```

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.

```
1717  \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1718    {
1719      \str_case:nnF {#1}
1720        {
1721          { series } {} { resume } {} { start } {}
1722          { start* } {}  { save-ans } {} { save-key } {}
1723        }
1724        { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1725    }
```

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

__enumext_parse_series:n
__enumext_resume_last:n

The function __enumext_parse_series:n will be responsible for storing the filtered ⟨keys⟩ in the global variable \g__enumext_series_⟨series name⟩_tl along with the creation of the integer variable \g__-enumext_series_⟨series name⟩_int when the key is passed as an argument; otherwise, it will check the state of the boolean variable \l__enumext_resume_active_bool set by the keys resume and resume* and will call the function __enumext_resume_last:n.

🌶 The value of boolean variable \l__enumext_resume_active_bool is set to true by the function __enumext_resume_-counter:n which is used by the keys resume and resume*, in this case we must Make sure it is set to false so that it does not overwrite the default filtered ⟨keys⟩. This function is passed to the function __enumext_parse_keys:n in the enumext environment definition (§12.38) and to the function __enumext_parse_keys_vii:n in the enumext* environment definition (§12.43).

```
1726  \cs_new_protected:Npn \__enumext_parse_series:n #1
1727    {
1728      \str_if_empty:NTF \l__enumext_series_str
1729        {
1730          \bool_if:NF \l__enumext_resume_active_bool
1731            {
1732              \__enumext_resume_last:n {#1}
1733            }
1734        }
1735        {
1736          \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str _tl }
1737          \tl_gset:ce { g__enumext_series_ \l__enumext_series_str _tl }
1738            { \__enumext_filter_series:n {#1} }
1739          \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str _int }
1740            {
1741              \int_new:c { g__enumext_series_ \l__enumext_series_str _int }
1742            }
1743        }
1744    }
```

The function __enumext_resume_last:n will be in charge of saving the filtering ⟨keys⟩ when the series key is *not used* and will save them in the variable \g__enumext_standar_series_tl for the enumext environment and in the variable \g__enumext_starred_series_tl for the enumext* environment. Here we must use \bool_lazy_all:nT to make sure that the default values are not overwritten when the environment is nested and the series key is not being used.

```
1745  \cs_new_protected:Npn \__enumext_resume_last:n #1
1746    {
1747      \bool_if:NT \l__enumext_standar_first_bool
1748        {
1749          \tl_gclear:N \g__enumext_standar_series_tl
1750          \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1751        }
1752      \bool_if:NT \l__enumext_starred_first_bool
1753        {
1754          \tl_gclear:N \g__enumext_starred_series_tl
1755          \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1756        }
1757    }
```

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

12.24.2 Internal function to save counter value

__enumext_resume_save_counter:

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

- The variables \l__enumext_series_str and \l__enumext__resume_name_tl contain the same {⟨series name⟩} but are executed at different moments, the integer variable with \l__enumext_series_str sets the value when execute series={⟨series name⟩} and the integer variable with \l__enumext__resume_name_tl sets the subsequent values when use resume={⟨series name⟩}. This function is passed to the enumext environment definition (§12.38) and the enumext* environment definition (§12.43).

```
1758  \cs_new_protected:Nn \__enumext_resume_save_counter:
1759    {
1760      \bool_if:NT \g__enumext_standar_bool
1761        {
1762          \tl_if_empty:NF \l__enumext_series_str
1763            {
1764              \int_gset_eq:cN
1765                { g__enumext_series_ \l__enumext_series_str _int } \value{enumXi}
1766            }
1767          \tl_if_empty:NTF \l__enumext_resume_name_tl
1768            {
1769              \str_if_empty:NT \l__enumext_series_str
1770                {
1771                  \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1772                }
1773            }
1774            {
1775              \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1776                {
1777                  \int_gset_eq:cN
1778                    { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXi}
1779                }
1780            }
1781          \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1782            {
1783              \int_gset_eq:cN
1784                { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1785            }
1786        }
1787      \bool_if:NT \g__enumext_starred_bool
1788        {
1789          \tl_if_empty:NF \l__enumext_series_str
1790            {
1791              \int_gset_eq:cN
1792                { g__enumext_series_ \l__enumext_series_str _int } \value{enumXvii}
1793            }
1794          \tl_if_empty:NTF \l__enumext_resume_name_tl
1795            {
1796              \str_if_empty:NT \l__enumext_series_str
1797                {
1798                  \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1799                }
1800            }
1801            {
1802              \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1803                {
1804                  \int_gset_eq:cN
1805                    { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXvii}
1806                }
1807            }
1808          \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1809            {
1810              \int_gset_eq:cN
1811                { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1812            }
1813        }
1814    }
```

(*End of definition for* __enumext_resume_save_counter:.)

12.24.3   Internal functions for resume key

__enumext_resume_series:n

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

```
1815 \cs_new_protected:Npn \__enumext_resume_series:n #1
1816   {
1817     \tl_if_empty:nTF {#1}
1818       {
1819         \__enumext_resume_counter:n { }
1820       }
1821       {
1822         \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1823           {
1824             \__enumext_resume_counter:n {#1}
1825             \bool_if:NT \g__enumext_standar_bool
1826               {
1827                 \keys_set:nv { enumext / level-1 }
1828                   { g__enumext_series_ \tl_to_str:n {#1} _tl }
1829               }
1830             \bool_if:NT \g__enumext_starred_bool
1831               {
1832                 \keys_set:nv { enumext / enumext* }
1833                   { g__enumext_series_ \tl_to_str:n {#1} _tl }
1834               }
1835           }
1836           {
1837             \bool_if:NT \g__enumext_standar_bool
1838               {
1839                 \msg_error:nnn { enumext } { unknown-series } {#1}
1840               }
1841             \bool_if:NT \g__enumext_starred_bool
1842               {
1843                 \msg_error:nnn { enumext } { unknown-series } {#1}
1844               }
1845           }
1846       }
1847   }
```

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

__enumext_resume_counter:n
__enumext_resume_counter:
__enumext_resume_counter_series:
__enumext_resume_counter_save_ans:

The function __enumext_resume_counter:n will set the variable \l__enumext_resume_active_bool to true and pass the value of the key resume to the variable \l__enumext_series_name_tl which will contain the {⟨series name⟩}. If the variable \l__enumext_series_name_tl is empty, that is, we are passing the key resume *without value*, we will execute the function __enumext_resume_counter: otherwise, when we pass resume={⟨series name⟩} we will execute the function __enumext_resume_counter_series:, finally we will execute the function __enumext_resume_counter_save_ans: which is associated with the key save-ans.

```
1848 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1849   {
1850     \bool_set_true:N \l__enumext_resume_active_bool
1851     \tl_set:Nn \l__enumext_resume_name_tl {#1}
1852     \tl_if_empty:NTF \l__enumext_resume_name_tl
1853       {
1854         \__enumext_resume_counter:
1855       }
1856       {
1857         \__enumext_resume_counter_series:
1858       }
1859     \__enumext_resume_counter_save_ans:
1860   }
```

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

```
1861 \cs_new_protected:Nn \__enumext_resume_counter:
1862   {
1863     \bool_if:NT \g__enumext_standar_bool
```

```
1864        {
1865            \int_gincr:N \g__enumext_resume_int
1866            \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1867        }
1868     \bool_if:NT \g__enumext_starred_bool
1869        {
1870            \int_gincr:N \g__enumext_resume_vii_int
1871            \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1872        }
1873   }
```

The function __enumext_resume_counter_series: will be executed when the resume={⟨series name⟩} key is active, setting the counters for the *"first level"* of the environments according to the value of the integer variables created by the series key.

```
1874  \cs_new_protected:Nn \__enumext_resume_counter_series:
1875    {
1876      \bool_if:NT \g__enumext_standar_bool
1877        {
1878          \int_set:Nn \l__enumext_start_i_int
1879            {
1880              \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1881            }
1882        }
1883      \bool_if:NT \g__enumext_starred_bool
1884        {
1885          \int_set:Nn \l__enumext_start_vii_int
1886            {
1887              \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1888            }
1889        }
1890    }
```

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

```
1891  \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1892    {
1893      \bool_lazy_and:nnT
1894        { \bool_if_p:N \l__enumext_standar_first_bool }
1895        { \bool_if_p:N \l__enumext_store_active_bool }
1896        {
1897          \int_set:Nn \l__enumext_start_i_int
1898            {
1899              \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1900            }
1901        }
1902      \bool_lazy_and:nnT
1903        { \bool_if_p:N \l__enumext_starred_first_bool }
1904        { \bool_if_p:N \l__enumext_store_active_bool }
1905        {
1906          \int_set:Nn \l__enumext_start_vii_int
1907            {
1908              \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1909            }
1910        }
1911    }
```

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

12.24.4   Internal function for resume* key

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

```
1912  \cs_new_protected:Nn \__enumext_resume_starred:
1913    {
1914      \bool_if:NT \g__enumext_standar_bool
1915        {
1916          \tl_if_empty:NF \g__enumext_standar_series_tl
1917            {
1918              \__enumext_resume_counter:n { }
```

```
1919              \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1920            }
1921      }
1922    \bool_if:NT \g__enumext_starred_bool
1923      {
1924        \tl_if_empty:NF \g__enumext_starred_series_tl
1925          {
1926            \__enumext_resume_counter:n { }
1927            \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1928          }
1929      }
1930  }
```

(*End of definition for* __enumext_resume_starred:.)

12.25   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.

12.25.1   Setting save-ans key

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

```
1931 \cs_set_protected:Npn \__enumext_tmp:n #1
1932   {
1933     \keys_define:nn { enumext / #1 }
1934       {
1935         save-ans .code:n = \__enumext_storing_set:n {##1},
1936         save-ans .value_required:n = true,
1937       }
1938   }
1939 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

12.25.2   Internal functions for save-ans key

__enumext_start_save_ans_msg: The functions __enumext_start_save_ans_msg: and __enumext_stop_save_ans_msg: will display
__enumext_stop_save_ans_msg: 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:.

```
1940 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
1941   {
1942     \msg_term:nnVV { enumext } { save-ans-log }
1943        \g__enumext_envir_name_tl \l__enumext_store_name_tl
1944   }
1945 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
1946   {
1947     \msg_term:nnVV { enumext } { save-ans-log-hook }
1948        \g__enumext_envir_name_tl \g__enumext_store_name_tl
1949   }
```

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

__enumext_storing_set:n   The function __enumext_storing_set:n first pass the value of the save-ans key to the variable \l__-
__enumext_storing_exec: 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.

```
1950 \cs_new_protected:Npn \__enumext_storing_set:n #1
1951   {
1952     \tl_set:Ne \l__enumext_store_name_tl {#1}
1953     \tl_if_empty:NTF \l__enumext_store_name_tl
1954       {
1955         \bool_lazy_or:nnT
1956           { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1957           {
1958             \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
1959           }
1960       }
1961       {
1962         \bool_lazy_or:nnT
```

```
1963            { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1964            {
1965              \__enumext_start_save_ans_msg:
1966              \__enumext_storing_exec:
1967            }
1968          }
1969      }
```

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 `keyans`, `keyans*` and `keyanspic` environments and will set to true the variable `\l__enumext_check_answers_bool` used for checking answers by the `check-ans` and `no-store` keys, copy {⟨*store name*⟩} into the global variable `\g__enumext_store_name_-tl` and execute the function `\__enumext_anskey_env_make:V` creating the environment `anskey*` (§12.30). 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*`.

```
1970  \cs_new_protected:Nn \__enumext_storing_exec:
1971    {
1972      \bool_set_true:N \l__enumext_store_active_bool
1973      \bool_set_true:N \l__enumext_check_answers_bool
1974      \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1975      \__enumext_anskey_env_make:V \l__enumext_store_name_tl
1976      \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1977        {
1978          \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
1979          \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1980        }
1981      \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1982        {
1983          \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
1984          \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1985        }
1986      \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1987        {
1988          \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
1989          \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1990        }
1991    }
```

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

12.25.3 The check answer mechanism

The mechanism for checking that all questions are answered 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.

12.25.4 Setting check-ans and no-store keys

<div style="text-align:right">check-ans
no-store</div>

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

```
1992 \cs_set_protected:Npn \__enumext_tmp:n #1
1993   {
1994     \keys_define:nn { enumext / #1 }
1995       {
1996         check-ans .bool_set:N = \l__enumext_check_ans_key_bool,
1997         check-ans .initial:n  = false,
1998         check-ans .value_required:n = true,
1999         no-store  .code:n = {
2000                             \bool_set_false:N \l__enumext_check_answers_bool
2001                             \bool_set_false:N \l__enumext_check_ans_key_bool
2002                           },
2003         no-store  .value_forbidden:n = true,
2004       }
2005   }
2006 \clist_map_inline:nn
2007   {
2008     level-1, level-2, level-3, level-4, enumext*
2009   }
2010   { \__enumext_tmp:n {#1} }
```

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

12.25.5 Set-up check answer mechanism

<div style="text-align:right">__enumext_check_ans_active:
__enumext_check_ans_level:</div>

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.

```
2011 \cs_new_protected:Nn \__enumext_check_ans_active:
2012   {
2013     \tl_if_empty:NF \l__enumext_store_name_tl
2014       {
2015         \bool_if:NT \l__enumext_check_answers_bool
2016           {
2017             \__enumext_check_ans_level:
2018           }
2019       }
2020   }
```

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"* .

```
2021 \cs_new_protected:Nn \__enumext_check_ans_level:
2022   {
2023     \int_case:nn { \l__enumext_level_int }
2024       {
2025         { 1 }{
2026             \bool_lazy_all:nT
2027               {
2028                 { \bool_if_p:N \g__enumext_starred_bool }
2029                 { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
2030               }
2031               {
2032                 \int_gdecr:N \g__enumext_item_number_int
2033                 \bool_set_false:N \l__enumext_item_number_bool
2034               }
2035           }
2036         { 2 }{
2037             \int_gdecr:N \g__enumext_item_number_int
2038             \bool_set_false:N \l__enumext_item_number_bool
2039           }
2040         { 3 }{
2041             \int_gdecr:N \g__enumext_item_number_int
2042             \bool_set_false:N \l__enumext_item_number_bool
2043           }
2044         { 4 }{
2045             \int_gdecr:N \g__enumext_item_number_int
2046             \bool_set_false:N \l__enumext_item_number_bool
```

```
2047                    }
2048              }
```

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.

```
2049        \int_case:nn { \l__enumext_level_h_int }
2050          {
2051            { 1 }{
2052                  \bool_lazy_all:nT
2053                    {
2054                      { \bool_if_p:N \g__enumext_standar_bool }
2055                      { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
2056                    }
2057                    {
2058                      \int_gdecr:N \g__enumext_item_number_int
2059                      \bool_set_false:N \l__enumext_item_number_bool
2060                    }
2061                }
2062          }
2063      }
```

(*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.

```
2064  \cs_new_protected:Nn \__enumext_check_ans_key_hook:
2065    {
2066      \bool_lazy_and:nnT
2067        { \bool_if_p:N \l__enumext_check_ans_key_bool }
2068        { \bool_if_p:N \g__enumext_standar_bool }
2069        {
2070          \bool_gset_true:N \g__enumext_check_ans_key_bool
2071        }
2072      \bool_lazy_and:nnT
2073        { \bool_if_p:N \l__enumext_check_ans_key_bool }
2074        { \bool_if_p:N \g__enumext_starred_bool }
2075        {
2076          \bool_gset_true:N \g__enumext_check_ans_key_bool
2077        }
2078    }
```

(*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:.

```
2079  \cs_new_protected:Nn \__enumext_item_answer_diff:
2080    {
2081      \int_gset:Nn \g__enumext_item_answer_diff_int
2082        {
2083          \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
2084        }
2085    }
```

(*End of definition for* __enumext_item_answer_diff:.)

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

```
2086  \cs_new_protected:Nn \__enumext_check_ans_show:
2087    {
2088      \int_case:nn { \g__enumext_item_answer_diff_int }
2089        {
2090          { -1 }{ \__enumext_check_ans_msg_less:     }
2091          {  0 }{ \__enumext_check_ans_msg_same_ok: }
2092          {  1 }{ \__enumext_check_ans_msg_greater: }
2093        }
2094    }
```

```
2095  \cs_new_protected:Nn \__enumext_check_ans_msg_less:
2096    {
2097      \msg_warning:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2098        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2099    }
2100  \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
2101    {
2102      \msg_term:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2103        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2104    }
2105  \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
2106    {
2107      \msg_warning:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2108        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2109    }
```

(*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:`.

```
2110  \cs_new_protected:Nn \__enumext_check_ans_log:
2111    {
2112      \int_case:nn { \g__enumext_item_answer_diff_int }
2113        {
2114          { -1 }{ \__enumext_check_ans_log_msg_less:    }
2115          {  0 }{ \__enumext_check_ans_log_msg_same_ok: }
2116          {  1 }{ \__enumext_check_ans_log_msg_greater: }
2117        }
2118    }
2119  \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
2120    {
2121      \msg_log:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2122        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2123    }
2124  \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
2125    {
2126      \msg_log:nneee { enumext } { items-same-answer }  { \g__enumext_store_name_tl }
2127        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2128    }
2129  \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
2130    {
2131      \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2132        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2133    }
```

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

12.25.6 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.

```
2134  \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
2135    {
2136      \int_compare:nNnT
2137        { \g__enumext_check_starred_cmd_int } = { 0 }
2138        {
2139          \msg_warning:nnnV
2140            { enumext } { missing-starred }{ #1 } \l__enumext_check_start_line_env_tl
2141        }
2142      \int_compare:nNnT
2143        { \g__enumext_check_starred_cmd_int } > { 1 }
2144        {
2145          \msg_warning:nnnV
2146            { enumext } { many-starred }{ #1 } \l__enumext_check_start_line_env_tl
2147        }
2148      \int_gzero:N \g__enumext_check_starred_cmd_int
2149      \tl_clear:N \l__enumext_check_start_line_env_tl
2150    }
```

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

12.26   Keys and functions associated with storage

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

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

```
2151 \cs_set_protected:Npn \__enumext_tmp:n #1
2152   {
2153     \keys_define:nn { enumext / #1 }
2154       {
2155         wrap-ans   .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
2156         wrap-ans   .initial:n =
2157                      {
2158                        \fbox{\parbox[t]{\dimeval{\itemwidth -2\fboxsep -2\fboxrule}}{##1}}
2159                      },
2160         wrap-ans   .value_required:n = true,
2161         wrap-opt   .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
2162         wrap-opt   .initial:n = [{##1}],
2163         wrap-opt   .value_required:n = true,
2164         save-sep   .tl_set:N  = \l__enumext_store_keyans_item_opt_sep_tl,
2165         save-sep   .initial:n = {, ~ },
2166         save-sep   .value_required:n = true,
2167         mark-ans   .tl_set:N  = \l__enumext_mark_answer_sym_tl,
2168         mark-ans   .initial:n = \textasteriskcentered,
2169         mark-ans   .value_required:n = true,
2170         mark-pos   .choice:,
2171         mark-pos / left     .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2172         mark-pos / right    .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2173         mark-pos / unknown .code:n =
2174                         \msg_error:nneee { enumext } { unknown-choice }
2175                           { mark-pos } { left, ~ right } { \exp_not:n {##1} },
2176         mark-pos   .initial:n = right,
2177         mark-pos   .value_required:n = true,
2178         show-ans   .bool_set:N = \l__enumext_show_answer_bool,
2179         show-ans   .initial:n  = false,
2180         show-ans   .value_required:n = true,
2181         show-pos   .bool_set:N = \l__enumext_show_position_bool,
2182         show-pos   .initial:n  = false,
2183         show-pos   .value_required:n = true,
2184         mark-ref   .tl_set:N   = \l__enumext_mark_ref_sym_tl,
2185         mark-ref   .initial:n  = \textasteriskcentered,
2186         mark-ref   .value_required:n = true,
2187         save-ref   .bool_set:N = \l__enumext_store_ref_key_bool,
2188         save-ref   .initial:n  = false,
2189         save-ref   .value_required:n = true,
2190       }
2191   }
2192 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

mark-pos
show-ans
show-pos

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

```
2193 \cs_set_protected:Npn \__enumext_tmp:n #1
2194   {
2195     \keys_define:nn { enumext / #1 }
2196       {
2197         mark-pos .choice:,
2198         mark-pos / left  .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2199         mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2200         mark-pos .initial:n = right,
2201         mark-pos .value_required:n  = true,
2202         show-ans .bool_set:N = \l__enumext_show_answer_bool,
2203         show-ans .initial:n  = false,
2204         show-ans .value_required:n = true,
2205         show-pos .bool_set:N = \l__enumext_show_position_bool,
2206         show-pos .initial:n  = false,
2207         show-pos .value_required:n = true,
2208       }
2209   }
2210 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }
```

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

12.26.1 Store optional arguments of the environments

The idea behind *"storing"* 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 arguments passed to the levels of the environment in which it is executed and *"storing"* them.

`\__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 *"storing"* the ⟨*keys*⟩ filtered from the optional arguments 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*⟩.

```
2211 \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2212   {
2213     \bool_if:cF { l__enumext_store_save_key_ \__enumext_level: _bool }
2214       {
2215         \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2216         \tl_set:ce
2217           { l__enumext_store_save_key_ \__enumext_level: _tl }
2218           { \__enumext_filter_save_key:n {#1} }
2219       }
2220   }
2221 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2222   {
2223     \bool_if:NF \l__enumext_store_save_key_vii_bool
2224       {
2225         \tl_clear:N \l__enumext_store_save_key_vii_tl
2226         \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2227       }
2228   }
```

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

12.26.2 Setting save-key key

Since this list structure will be stored in the ⟨*sequence*⟩ established by the `save-ans` key when executing `\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 list stored in the ⟨*sequence*⟩.

`save-key`

The values set by this key passed in the optional arguments 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`.

Define the key `save-key` for all levels of `enumext` and `enumext*` environments.

```
2229 \cs_set_protected:Npn \__enumext_tmp:n #1
2230   {
2231     \keys_define:nn { enumext / enumext* }
2232       {
2233         save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2234         save-key .value_required:n = true,
2235       }
2236     \keys_define:nn { enumext / #1 }
2237       {
2238         save-key .code:n = \__enumext_parse_save_key:n {##1},
2239         save-key .value_required:n = true,
2240       }
2241   }
2242 \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 the filtered ⟨*keys*⟩ in the variable `\l__enumext_store_save_key_X_tl` for `enumext` and `enumext*`.

```
2243 \cs_new_protected:Npn \__enumext_parse_save_key:n #1
2244   {
2245     \bool_set_true:c { l__enumext_store_save_key_ \__enumext_level: _bool }
2246     \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2247     \tl_set:ce
2248       { l__enumext_store_save_key_ \__enumext_level: _tl }
2249       { \__enumext_filter_save_key:n {#1} }
2250   }
```

```
2251 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2252   {
2253     \bool_set_true:N \l__enumext_store_save_key_vii_bool
2254     \tl_clear:N \l__enumext_store_save_key_vii_tl
2255     \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2256   }
```

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

12.26.3 Internal functions to store optional arguments

__enumext_filter_save_key:n
__enumext_filter_save_key_key:n
__enumext_filter_save_key_pair:nn

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

```
2257 \cs_new:Npn \__enumext_filter_save_key:n #1
2258   {
2259     \use:e
2260       {
2261         \keyval_parse:NNn
2262           \__enumext_filter_save_key_key:n
2263           \__enumext_filter_save_key_pair:nn {#1}
2264       }
2265   }
```

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

```
2266 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2267   {
2268     \str_case:nnF {#1}
2269       {
2270         { resume } {} { resume* } {} { no-store } {} { base-fix } {}
2271       }
2272       { , { \exp_not:n {#1} } }
2273   }
```

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

```
2274 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2275   {
2276     \str_case:nnF {#1}
2277       {
2278         { series   } {} { resume    } {} { save-ans } {} { save-ref   } {}
2279         { save-key } {} { check-ans } {} { show-ans } {} { show-pos   } {}
2280         { wrap-ans } {} { mark-ans  } {} { wrap-opt } {} { save-sep   } {}
2281         { mark-ref } {} { mini-env  } {} { mini-sep } {} { mini-right } {}
2282         { mini-right* } {}
2283       }
2284       { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2285   }
```

(*End of definition for* __enumext_filter_save_key:n, __enumext_filter_save_key_key:n, *and* __enumext_filter_-save_key_pair:nn.)

12.26.4 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.

```
2286 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2287   {
2288     \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2289       {
2290         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2291       }
2292       { #1 }
2293   }
2294 \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V, e }
```

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

12.26.5 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.

```
2295  \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2296    {
2297      \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
2298    }
2299  \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V, e }
```

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

12.26.6 Functions for storing the list structure in the sequence

__enumext_store_level_open:
__enumext_store_level_close:

The memorization structure of the list is handled by the functions __enumext_store_level_open: and __enumext_store_level_close: which are executed per level within the enumext environment.

```
2300  \cs_new_protected:Nn \__enumext_store_level_open:
2301    {
2302      \bool_if:NT \l__enumext_check_answers_bool
2303        {
2304          \tl_if_empty:cTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2305            {
2306              \__enumext_store_addto_seq:n
2307                {
2308                  \item \begin{enumext}
2309                }
2310            }
2311            {
2312              \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2313                {
2314                  \item \begin{enumext} [
2315                }
2316              \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2317                {
2318                  ]
2319                }
2320              \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2321            }
2322        }
2323    }
2324  \cs_new_protected:Nn \__enumext_store_level_close:
2325    {
2326      \bool_if:NT \l__enumext_check_answers_bool
2327        {
2328          \__enumext_store_addto_seq:n { \end{enumext} }
2329        }
2330    }
```

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

__enumext_store_level_open_vii:
__enumext_store_level_close_vii:

The memorization structure of the list is handled by the functions __enumext_store_level_open_vii: and __enumext_store_level_close_vii: which are executed in the enumext* environment.

```
2331  \cs_new_protected:Nn \__enumext_store_level_open_vii:
2332    {
2333      \bool_if:NT \l__enumext_check_answers_bool
2334        {
2335          \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2336            {
2337              \__enumext_store_addto_seq:n
2338                {
2339                  \item \begin{enumext*}
2340                }
2341            }
2342            {
2343              \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2344                {
2345                  \item \begin{enumext*}[
2346                }
2347              \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
```

```
2348                    {
2349                      ]
2350                    }
2351             \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2352           }
2353        }
2354    }
2355 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2356    {
2357      \bool_if:NT \l__enumext_check_answers_bool
2358        {
2359          \__enumext_store_addto_seq:n { \end{enumext*} }
2360        }
2361    }
```

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

12.26.7 Function for show marks and position

__enumext_print_keyans_box:NN
__enumext_print_keyans_box:cc

The function __enumext_print_keyans_box:NN print a box in the left margin with \l__enumext_mark_-answer_sym_tl used by the wrap-ans, show-ans and show-pos keys. The function takes two arguments:

#1: \l__enumext_labelwidth_X_dim
#2: \l__enumext_labelsep_X_dim

```
2362 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2363    {
2364      \mode_leave_vertical:
2365      \skip_horizontal:n { -\dim_use:N #2 }
2366      \makebox[0pt][ r ]
2367        {
2368          \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2369            {
2370              \tl_use:N \l__enumext_mark_answer_sym_tl
2371            }
2372        }
2373      \skip_horizontal:n { \dim_use:N #2 }
2374    }
2375 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }
```

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

12.27 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.

```
2376 \cs_new_protected:Nn \__enumext_store_internal_ref:
2377    {
2378      \cs_set_protected:Npn \__enumext_tmp:n ##1
2379        {
2380          \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2381          \tl_reverse:c { l__enumext_label_copy_##1_tl }
2382          \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2383          \tl_reverse:c { l__enumext_label_copy_##1_tl }
2384        }
2385      \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2386      \cs_set:Npn \__enumext_tmp:n ##1
2387        { . \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.

```
2388        \bool_lazy_all:nT
2389          {
2390            { \bool_if_p:N \g__enumext_starred_bool }
2391            { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2392          }
2393          {
2394            \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2395              { \tl_use:N \l__enumext_label_copy_vii_tl }
2396          }
```

```
2397        \bool_lazy_all:nT
2398          {
2399            { \bool_not_p:n { \g__enumext_standar_bool } }
2400            { \bool_if_p:N \l__enumext_standar_bool }
2401            { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2402          }
2403          {
2404            \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2405              {
2406                \tl_use:N \l__enumext_label_copy_vii_tl
2407                \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2408              }
2409          }
```

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.

```
2410        \bool_lazy_all:nT
2411          {
2412            { \bool_if_p:N \g__enumext_standar_bool }
2413            { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2414            { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2415          }
2416          {
2417            \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2418              {
2419                \tl_use:N \l__enumext_label_copy_i_tl
2420                \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2421              }
2422          }
2423        \cs_set:Npn \__enumext_tmp:n ##1
2424          { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } . }
2425        \bool_lazy_all:nT
2426          {
2427            { \bool_if_p:N \g__enumext_standar_bool }
2428            { \bool_if_p:N \l__enumext_starred_bool }
2429            { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2430          }
2431          {
2432            \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2433              {
2434                \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2435                \tl_use:N \l__enumext_label_copy_vii_tl
2436              }
2437          }
```

Now we set the variable \l__enumext_newlabel_arg_one_tl which will contain {⟨*store name* : *position*⟩}.

```
2438        \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2439          {
2440            \l__enumext_store_name_tl \c_colon_str
2441            \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2442          }
```

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.

```
2443        \tl_put_right:Ne \l__enumext_write_aux_file_tl
2444          {
2445            \__enumext_newlabel:nn
2446              { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2447              { \l__enumext_newlabel_arg_two_tl }
2448          }
2449        \l__enumext_write_aux_file_tl
2450      }
```

(*End of definition for* __enumext_store_internal_ref:.)

12.28 Common functions for \anskey and anskey* environment

__enumext_store_anskey_code:n

The internal function __enumext_store_anskey_code: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*⟩.

```
2451  \cs_new_protected:Npn \__enumext_store_anskey_code:n #1
2452    {
```

```
2453    \int_gincr:N \g__enumext_item_anskey_int
2454    \__enumext_store_addto_prop:n {#1}
2455    \bool_if:NT \l__enumext_store_ref_key_bool
2456      {
2457        \__enumext_store_internal_ref:
2458      }
2459    \__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.

```
2460    \tl_clear:N \l__enumext_store_anskey_arg_tl
2461    \bool_lazy_and:nnT
2462      { \bool_if_p:N \l__enumext_store_columns_break_bool }
2463      { \bool_not_p:n { \l__enumext_starred_bool } }
2464      {
2465        \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2466      }
2467    \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.

```
2468    \bool_lazy_and:nnT
2469      { \bool_not_p:n { \l__enumext_starred_bool } }
2470      { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2471      {
2472        \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2473          {
2474            ( \exp_not:V \l__enumext_store_item_join_int )
2475          }
2476      }
```

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*.

```
2477    \bool_if:NTF \l__enumext_store_item_star_bool
2478      {
2479        \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2480        \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2481          {
2482            \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2483              {
2484                [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2485              }
2486          }
2487        \dim_compare:nT
2488          {
2489            \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2490          }
2491          {
2492            \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2493              {
2494                [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2495              }
2496          }
2497        \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2498      }
2499      {
2500        \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2501      }
```

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⟩.

```
2502    \bool_lazy_and:nnT
2503      { \bool_if_p:N \l__enumext_store_ref_key_bool }
2504      { \bool_if_p:N \l__enumext_hyperref_bool }
2505      {
2506        \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2507          {
2508            \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2509              { \exp_not:V \l__enumext_mark_ref_sym_tl }
2510          }
```

```
2511        }
2512        \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2513    }
```

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

__enumext_anskey_show_wrap_arg:n The function __enumext_anskey_show_wrap_arg:n *"wraps"* the ⟨*argument*⟩ passed to \anskey and the ⟨*body*⟩ for anskey* when using the wrap-ans key.

```
2514  \cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
2515    {
2516      \par
2517      \bool_if:NTF \l__enumext_starred_bool
2518        {
2519          \__enumext_print_keyans_box:NN \l__enumext_labelwidth_vii_dim  \l__enumext_labelsep_vii_d
2520        }
2521        {
2522          \__enumext_print_keyans_box:cc
2523            { l__enumext_labelwidth_ \__enumext_level: _dim }
2524            { l__enumext_labelsep_ \__enumext_level: _dim }
2525        }
2526      \__enumext_anskey_wrapper:n { #1 }
2527    }
```

(*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.

```
2528  \cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
2529    {
2530      \bool_if:NT \l__enumext_show_answer_bool
2531        {
2532          \__enumext_anskey_show_wrap_arg:n { #1 }
2533        }
2534      \bool_if:NT \l__enumext_show_position_bool
2535        {
2536          \tl_set:Ne \l__enumext_mark_answer_sym_tl
2537            {
2538              \group_begin:
2539              \exp_not:N \normalfont
2540              \exp_not:N \footnotesize [ \int_eval:n
2541                {
2542                  \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2543                }
2544                ]
2545              \group_end:
2546            }
2547          \__enumext_anskey_show_wrap_arg:n { #1 }
2548        }
2549    }
```

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

12.29   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*⟩}.

__enumext_anskey_unknown:n   First we'll add the keys break-col, item-join, item-star, item-sym* and item-pos*.
__enumext_anskey_unknown:nn

```
2550  \keys_define:nn { enumext / anskey }
2551    {
2552      break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2553      break-col .default:n  = true,
2554      break-col .value_forbidden:n = true,
2555      item-join .int_set:N  = \l__enumext_store_item_join_int,
2556      item-join .value_required:n = true,
2557      item-star .bool_set:N = \l__enumext_store_item_star_bool,
2558      item-star .default:n  = true,
```

```
2559        item-star .value_forbidden:n = true,
2560        item-sym* .tl_set:N     = \l__enumext_store_item_symbol_tl,
2561        item-sym* .value_required:n = true,
2562        item-pos* .dim_set:N    = \l__enumext_store_item_symbol_sep_dim,
2563        item-pos* .value_required:n = true,
2564        unknown   .code:n       = { \__enumext_anskey_unknown:n {#1} },
2565      }
```

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.

```
2566  \cs_new_protected:Npn \__enumext_anskey_unknown:n #1
2567    {
2568      \exp_args:NV \__enumext_anskey_unknown:nn \l_keys_key_str {#1}
2569    }
2570  \cs_new_protected:Npn \__enumext_anskey_unknown:nn #1 #2
2571    {
2572      \tl_if_blank:nTF {#2}
2573        {
2574          \msg_error:nnn { enumext } { anskey-cmd-key-unknown } {#1}
2575        }
2576        {
2577          \msg_error:nnnn { enumext } { anskey-cmd-key-value-unknown } {#1} {#2}
2578        }
2579    }
```

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

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

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

```
2580  \NewDocumentCommand \anskey { o +m }
2581    {
2582      \__enumext_anskey_safe_outer:
2583      \group_begin:
2584        \bool_if:NT \l__enumext_check_answers_bool
2585          {
2586            \tl_if_novalue:nF {#1}
2587              {
2588                \keys_set:nn { enumext / anskey } {#1}
2589              }
2590            \tl_if_blank:nTF {#2}
2591              {
2592                \msg_error:nn { enumext } { anskey-empty-arg }
2593              }
2594              {
2595                \__enumext_anskey_safe_inner:
2596                \__enumext_store_anskey_code:n {#2}
2597              }
2598          }
2599      \group_end:
2600    }
```

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

12.29.1   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.

```
2601  \cs_new_protected:Nn \__enumext_anskey_safe_outer:
2602    {
2603      \bool_if:NF \l__enumext_store_active_bool
2604        {
2605          \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
2606        }
2607      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2608        {
2609          \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
```

```
2610        }
2611      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2612        {
2613          \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans* }
2614        }
2615      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2616        {
2617          \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
2618        }
2619    }
```

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.

```
2620  \cs_new_protected:Nn \__enumext_anskey_safe_inner:
2621    {
2622      \int_incr:N \l__enumext_anskey_level_int
2623      \int_compare:nNnT { \l__enumext_anskey_level_int } > { 1 }
2624        {
2625          \msg_error:nn { enumext } { anskey-nested }
2626        }
2627      \bool_if:NF \l__enumext_item_number_bool
2628        {
2629          \msg_error:nn { enumext } { anskey-unnumber-item }
2630        }
2631      \mode_if_math:T
2632        {
2633          \msg_error:nne { enumext } { anskey-math-mode } { \c_backslash_str anskey }
2634        }
2635    }
```

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

12.30 The environment anskey*

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

– If some package, class or user has defined the environment with the same name somewhere in the document it would be a problem, you would not know what argument has been passed to store-env, if you are using the key print-env or the write-out key, sure, I can detect and modify it within the enumext and enumext* environments, but it would look strange not to have some keys available when running within these environments.

– A better (perhaps a bit paranoid) option is to define it within the environment in which the save-ans key is executed. and have it available only when that key is executed, here I would have absolute control of the ⟨keys⟩ and I make sure that write-out is not used, then using *hooks after* I undefine it and using *hook before* I check if it has been created by any package, class or user and I return a error, then the user will have to see how to solve the problem.

__enumext_undefine_anskey_env: The function __enumext_undefine_anskey_env: will undefine the environment anskey* and will be passed to the function __enumext_execute_after_env: (§12.31) which is executed after the environment in which the key save-ans is active.

```
2636  \cs_new_protected:Nn \__enumext_undefine_anskey_env:
2637    {
2638      \cs_undefine:c { anskey* }
2639      \cs_undefine:c { endanskey* }
2640      \cs_undefine:c { __scontents_anskey*_env_begin: }
2641      \cs_undefine:c { __scontents_anskey*_env_end: }
2642    }
```

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

```
2643  \__enumext_before_env:nn { enumext }
2644    {
2645      \bool_lazy_and:nnT
2646        { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2647        { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2648        {
2649          \cs_if_free:cF { __scontents_anskey*_env_begin: }
2650            {
2651              \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2652            }
2653        }
```

```
2654    }
2655 \__enumext_before_env:nn { enumext* }
2656    {
2657      \bool_lazy_and:nnT
2658        { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2659        { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2660        {
2661          \cs_if_free:cF { __scontents_anskey*_env_begin: }
2662            {
2663              \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2664            }
2665        }
2666    }
```

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

```
2667 \__enumext_before_env:nn { anskey* }
2668    {
2669      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2670        {
2671          \msg_error:nnn { enumext } { anskey-env-wrong }{ keyans }
2672        }
2673      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2674        {
2675          \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
2676        }
2677      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2678        {
2679          \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
2680        }
2681      \bool_if:NF \l__enumext_item_number_bool
2682        {
2683          \msg_error:nn { enumext } { anskey-unnumber-item }
2684        }
2685      \mode_if_math:T
2686        {
2687          \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
2688        }
2689    }
```

(*End of definition for* __enumext_undefine_anskey_env:.)

anskey*
__enumext_anskey_env_make:n
__enumext_anskey_env_make:V
__enumext_anskey_env_define_keys:
__enumext_rescan_anskey_env:n

The function __enumext_anskey_env_make:n creates the environment anskey* (*custom version* of scontents environment) by setting the initial keys store-env={⟨*store name*⟩} and print-env=false. To maintain the *scope* of the environment and that it is only active when the key save-ans is active we will pass this function to the function __enumext_storing_exec: (§12.25.1) and we will execute it only if the variable \l__enumext_anskey_env_bool is true, with this we prevent it from being executed again when the environment is nested and the key save-ans is active, which returns an error for part of the package scontents.

```
2690 \cs_new_protected:Npn \__enumext_anskey_env_make:n #1
2691    {
2692      \bool_if:NT \l__enumext_anskey_env_bool
2693        {
2694          \newenvsc{anskey*}[store-env=#1,print-env=false]
2695          \__enumext_anskey_env_exec:
2696        }
2697    }
2698 \cs_generate_variant:Nn \__enumext_anskey_env_make:n { V }
```

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

```
2699 \cs_new_protected:Nn \__enumext_anskey_env_define_keys:
2700    {
2701      \keys_define:nn { scontents / scontents }
2702        {
2703          break-col .bool_gset:N = \g__enumext_store_columns_break_bool,
2704          break-col .default:n   = true,
2705          break-col .value_forbidden:n = true,
2706          item-join .int_gset:N  = \g__enumext_store_item_join_int,
2707          item-join .value_required:n = true,
```

```
2708        item-star .bool_gset:N = \g__enumext_store_item_star_bool,
2709        item-star .default:n    = true,
2710        item-star .value_forbidden:n = true,
2711        item-sym* .tl_gset:N    = \g__enumext_store_item_symbol_tl,
2712        item-sym* .value_required:n = true,
2713        item-pos* .dim_gset:N   = \g__enumext_store_item_symbol_sep_dim,
2714        item-pos* .value_required:n = true,
2715        print-env .undefine:,
2716        store-env .undefine:,
2717        write-out .undefine:,
2718        unknown   .code:n       = { \__enumext_anskey_env_unknown:n {##1} },
2719      }
2720    }
```

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.

```
2721 \cs_new_protected:Npn \__enumext_anskey_env_unknown:n #1
2722   {
2723     \exp_args:NV \__enumext_anskey_env_unknown:nn \l_keys_key_str {#1}
2724   }
2725 \cs_new_protected:Npn \__enumext_anskey_env_unknown:nn #1#2
2726   {
2727     \tl_if_blank:nTF {#2}
2728       {
2729         \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
2730       }
2731       {
2732         \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
2733       }
2734   }
```

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

```
2735 \cs_new_protected:Nn \__enumext_anskey_env_reset_keys:
2736   {
2737     \keys_define:nn { scontents / scontents }
2738       {
2739         break-col .undefine:,
2740         item-join .undefine:,
2741         item-star .undefine:,
2742         item-sym* .undefine:,
2743         item-pos* .undefine:,
2744         write-out .code:n     = {
2745                                   \bool_set_false:N \l__scontents_storing_bool
2746                                   \bool_set_true:N  \l__scontents_writing_bool
2747                                   \tl_set:Nn \l__scontents_fname_out_tl {##1}
2748                                 },
2749         write-out .value_required:n = true,
2750         print-env .meta:nn    = { scontents } { print-env = ##1 },
2751         print-env .default:n = true,
2752         store-env .meta:nn    = { scontents } { store-env = ##1 },
2753         unknown   .code:n     = { \__scontents_parse_environment_keys:n {##1} },
2754       }
2755   }
```

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

```
2756 \cs_new_protected:Npn \__enumext_rescan_anskey_env:n #1
2757   {
2758     \group_begin:
2759       \int_set:Nn \tex_newlinechar:D { `\^^J }
2760       \__scontents_rescan_tokens:x
2761         {
2762           \endgroup % This assumes \catcode`\\=0... Things might go off otherwise.
2763           #1
2764         }
2765   }
```

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

__enumext_anskey_env_exec:

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

```
2766 \cs_new_protected:Nn \__enumext_anskey_env_exec:
2767   {
2768     \__enumext_before_env:nn { anskey* }
2769       {
2770         \__enumext_anskey_env_define_keys:
2771       }
```

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

```
2772     \hook_if_empty:nF {env/anskey*/after}
2773       {
2774         \hook_gremove_code:nn {env/anskey*/after} { * }
2775       }
2776     \__enumext_after_env:nn { anskey* }
2777       {
2778         \__enumext_anskey_env_save_keys:
2779         \tl_clear:N \l__enumext_store_anskey_env_tl
2780         \tl_clear:N \l__enumext_store_anskey_opt_tl
2781         \bool_if:NT \l__enumext_check_answers_bool
2782           {
2783             \tl_gset:Ne \l__enumext_store_anskey_env_tl
2784               {
2785                 \seq_item:ce { g__scontents_name_ \l__enumext_store_name_tl _seq } { -1 }
2786               }
2787             \regex_match:nVTF
2788               { ^\s* \z | ^\s* \u{c__scontents_hidden_space_str} \z }
2789               \l__enumext_store_anskey_env_tl
2790               {
2791                 \msg_error:nn { enumext } { anskey-empty-arg }
2792               }
2793               {
2794                 \__enumext_anskey_env_store:
2795               }
2796           }
2797         \__enumext_anskey_env_clean_vars:
2798         \__enumext_anskey_env_reset_keys:
2799       }
2800   }
```

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

(*End of definition for* __enumext_anskey_env_exec:.)

__enumext_anskey_env_save_keys:
__enumext_anskey_env_store:
__enumext_anskey_env_clean_vars:

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

```
2801 \cs_new_protected:Nn \__enumext_anskey_env_save_keys:
2802   {
2803     \bool_lazy_and:nnT
2804       { \bool_if_p:N \g__enumext_store_columns_break_bool }
2805       { \bool_not_p:n { \l__enumext_starred_bool } }
2806       {
2807         \tl_put_left:Ne \l__enumext_store_anskey_opt_tl { ,break-col, }
2808       }
```

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

```
2809     \bool_lazy_and:nnT
2810       { \bool_not_p:n { \l__enumext_starred_bool } }
2811       { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }
2812       {
2813         \tl_put_left::Ne \l__enumext_store_anskey_opt_tl
2814           {
2815             ,item-join = \exp_not:V \g__enumext_store_item_join_int,
2816           }
2817       }
```

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

```
2818      \bool_if:NT \g__enumext_store_item_star_bool
2819        {
2820          \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2821            {
2822              ,item-star,
2823            }
2824          \tl_if_empty:NF \g__enumext_store_item_symbol_tl
2825            {
2826              \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2827                {
2828                  ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,
2829                }
2830            }
2831          \dim_compare:nT
2832            {
2833              \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
2834            }
2835            {
2836              \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2837                {
2838                  ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
2839                }
2840            }
2841        }
2842      }
```

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

```
2843  \cs_new_protected:Nn \__enumext_anskey_env_store:
2844    {
2845      \group_begin:
2846        \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
2847          {
2848            \exp_args:Ne
2849              \__enumext_store_anskey_code:n
2850                {
2851                  \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2852                }
2853          }
2854          {
2855            \keys_set_known:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
2856            \exp_args:Ne
2857              \__enumext_store_anskey_code:n
2858                {
2859                  \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2860                }
2861          }
2862      \group_end:
2863    }
```

The function `\__enumext_anskey_env_clean_vars:` will return the global variables used by the ⟨*keys*⟩ to their initial state.

```
2864  \cs_new_protected:Nn \__enumext_anskey_env_clean_vars:
2865    {
2866      \bool_gset_false:N \g__enumext_store_columns_break_bool
2867      \int_gzero:N       \g__enumext_store_item_join_int
2868      \bool_gset_false:N \g__enumext_store_item_star_bool
2869      \tl_gclear:N       \g__enumext_store_item_symbol_tl
2870      \dim_gzero:N       \g__enumext_store_item_symbol_sep_dim
2871    }
```

(*End of definition for* `\__enumext_anskey_env_save_keys:` , `\__enumext_anskey_env_store:` , *and* `\__enumext_anskey_env_-clean_vars:`.)

12.31 Executing anskey*, check-ans and write .log

`\__enumext_execute_after_env:`

The `\__enumext_execute_after_env:` function will first return the appropriate message for the end of the environment in which the `save-ans` key is being executed, then call the `\__enumext_item_answer_diff:` function and then will write the values of the global variables used to the `.log` file. If the key `check-ans` is active it will execute the function `\__enumext_check_ans_show:` and show the result in the terminal,

otherwise it will execute the function `\__enumext_check_ans_log:` and write the results in the `.log` file, undefine the environment anskey* (§12.30) through the function `\__enumext_undefine_anskey_env:` and finally we execute the function `\__enumext_reset_global_vars:` returning the used variables to their original state.

```
2872  \cs_new_protected:Nn \__enumext_execute_after_env:
2873    {
2874      \int_compare:nNnT { \l__enumext_level_int } = { 0 }
2875        {
2876          \tl_if_empty:NF \g__enumext_store_name_tl
2877            {
2878              \__enumext_stop_save_ans_msg:
2879              \__enumext_item_answer_diff:
2880              \__enumext_log_global_vars:
2881              \__enumext_log_answer_vars:
2882              \bool_if:NTF \g__enumext_check_ans_key_bool
2883                {
2884                  \__enumext_check_ans_show:
2885                }
2886                { \__enumext_check_ans_log: }
2887              \__enumext_undefine_anskey_env:
2888            }
2889          \__enumext_reset_global_vars:
2890        }
2891    }
```

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

💣 This function is passed to the function `\__enumext_after_env:nn` for the environments enumext (§12.38) and enumext* (§12.43) and it is executed only when the environments are not nested or at some level of these..

12.32 Common functions for keyans, keyans* and keyanspic

12.32.1 Storing content in prop list

`\__enumext_keyans_addto_prop:n`

The function `\__enumext_keyans_addto_prop:n` will pass the contents of the current ⟨*label*⟩ `\l__enumext_label_v_tl` for the keyans environment and the current ⟨*label*⟩ `\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, which will be passed to the ⟨*prop list*⟩ defined by the save-ans key using the `\__enumext_store_addto_prop:V`.

```
2892  \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2893    {
2894      \tl_clear:N \l__enumext_store_current_label_tl
2895      \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2896        {
2897          \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
2898        }
2899        {
2900          \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
2901        }
2902      \tl_if_novalue:nF { #1 }
2903        {
2904          % Set save-sep
2905          \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2906            {
2907              \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_o|
2908            }
2909          \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2910        }
2911      \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
2912    }
```

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

12.32.2 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 the `\anskey` command, basically because in this environments we are interested in the current ⟨*label*⟩. 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 our references.

```
2913  \cs_new_protected:Nn \__enumext_keyans_store_ref:
2914    {
2915      \bool_if:NT \l__enumext_store_ref_key_bool
2916        {
2917          \cs_set_protected:Npn \__enumext_tmp:n ##1
2918            {
2919              \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2920              \tl_reverse:c { l__enumext_label_copy_##1_tl }
2921              \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2922              \tl_reverse:c { l__enumext_label_copy_##1_tl }
2923            }
2924          \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2925          \__enumext_keyans_store_ref_aux_i:
2926        }
2927    }
```

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.

```
2928  \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2929    {
2930      \bool_if:NT \g__enumext_starred_bool
2931        {
2932          \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2933        }
2934      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2935        {
2936          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2937            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
2938        }
2939      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2940        {
2941          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2942            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2943        }
2944      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2945        {
2946          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2947            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2948        }
2949      \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2950        {
2951          \l__enumext_store_name_tl \c_colon_str
2952          \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2953        }
2954      \__enumext_keyans_store_ref_aux_ii:
2955    }
```

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.

```
2956  \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2957    {
2958      \tl_put_right:Ne \l__enumext_write_aux_file_tl
2959        {
2960          \__enumext_newlabel:nn
2961            { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2962            { \l__enumext_newlabel_arg_two_tl }
2963        }
2964      \l__enumext_write_aux_file_tl
2965    }
```

(*End of definition for* __enumext_keyans_store_ref:, __enumext_keyans_store_ref_aux_i:, *and* __enumext_keyans_-store_ref_aux_ii:.)

12.32.3   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.

```
2966  \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
```

```
2967    {
2968      \tl_clear:N \l__enumext_store_current_label_tl
2969      \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2970        {
2971          \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
2972        }
2973        {
2974          \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
2975        }
2976      \tl_if_novalue:nF { #1 }
2977        {
2978          \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2979            {
2980              \tl_put_right:Ne \l__enumext_store_current_label_tl
2981                {
2982                  \l__enumext_store_keyans_item_opt_sep_tl
2983                }
2984            }
2985          \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2986        }
2987      \__enumext_keyans_addto_seq_link:
2988    }
```

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.

```
2989  \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2990    {
2991      \bool_lazy_and:nnT
2992        { \bool_if_p:N \l__enumext_store_ref_key_bool }
2993        { \bool_if_p:N \l__enumext_hyperref_bool }
2994        {
2995          \tl_put_right:Ne \l__enumext_store_current_label_tl
2996            {
2997              \hfill \exp_not:N \hyperlink
2998                {
2999                  \exp_not:V \l__enumext_newlabel_arg_one_tl
3000                }
3001                { \exp_not:V \l__enumext_mark_ref_sym_tl }
3002            }
3003        }
3004      \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
3005      \bool_if:NT \l__enumext_check_answers_bool
3006        {
3007          \int_gincr:N \g__enumext_item_anskey_int
3008        }
3009    }
```

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

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

The code is very similar to the \anskey code, but, if I change the order of the operations the counter off ⟨*label*⟩ are incorrect.

__enumext_keyans_show_left:n
__enumext_keyans_show_ans:
__enumext_keyans_show_pos:
__enumext_keyans_show_item_opt:

Common function to show *starred commands* \item* and ⟨*position*⟩ of stored content in ⟨*prop list*⟩ for keyans and keyanspic. Need add 1 to \g__enumext_⟨*store name*⟩_prop for show-pos key.

```
3010  \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
3011    {
3012      \tl_if_novalue:nF { #1 }
3013        {
3014          \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
3015        }
3016      \bool_if:NT \l__enumext_show_answer_bool
3017        {
3018          \__enumext_keyans_show_ans:
3019        }
3020      \bool_if:NT \l__enumext_show_position_bool
3021        {
```

```
3022              \__enumext_keyans_show_pos:
3023            }
3024      }
3025  \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
3026      {
3027        \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
3028            {
3029              \bool_lazy_or:nnT
3030                { \bool_if_p:N \l__enumext_show_answer_bool }
3031                { \bool_if_p:N \l__enumext_show_position_bool }
3032                {
3033                  \__enumext_keyans_wrapper_opt:n { \l__enumext_store_current_opt_arg_tl } \c_space_tl
3034                }
3035            }
3036      }
3037  \cs_new_protected:Nn \__enumext_keyans_show_ans:
3038      {
3039        \bool_if:NT \l__enumext_starred_bool
3040            {
3041              \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
3042              \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
3043            }
3044        \tl_put_left:Nn \l__enumext_label_v_tl
3045            {
3046              \__enumext_print_keyans_box:NN
3047                \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3048            }
3049      }
3050  \cs_new_protected:Nn \__enumext_keyans_show_pos:
3051      {
3052        \bool_if:NT \l__enumext_starred_bool
3053            {
3054              \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
3055              \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
3056            }
3057        \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
3058            {
3059              \tl_set:Ne \l__enumext_mark_answer_sym_tl
3060                {
3061                  \group_begin:
3062                  \exp_not:N \normalfont
3063                  \exp_not:N \footnotesize [ \int_eval:n
3064                    {
3065                      \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3066                    }
3067                    ]
3068                  \group_end:
3069                }
3070            }
3071            {
3072              \tl_set:Ne \l__enumext_mark_answer_sym_tl
3073                {
3074                  \group_begin:
3075                  \exp_not:N \normalfont
3076                  \exp_not:N \footnotesize [ \int_eval:n
3077                    {
3078                      \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
3079                    }
3080                    ]
3081                  \group_end:
3082                }
3083            }
3084        \tl_put_left:Nn \l__enumext_label_v_tl
3085            {
3086              \__enumext_print_keyans_box:NN
3087                \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3088            }
3089      }
```

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

12.33 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.

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.

```
3090  \cs_new_protected:Npn \__enumext_default_item:n #1
3091    {
3092      \tl_if_novalue:nTF {#1}
3093        {
3094          \bool_if:NT \l__enumext_check_answers_bool
3095            {
3096              \int_gincr:N \g__enumext_item_number_int
3097              \bool_set_true:N \l__enumext_item_number_bool
3098            }
3099          \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
3100          \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3101        }
3102        {
3103          \bool_set_eq:cc
3104            { l__enumext_wrap_label_ \__enumext_level: _bool }
3105            { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
3106          \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl
3107        }
3108    }
```

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

__enumext_starred_item:nn   The \item*, \item*[⟨*symbol*⟩] and \item*[⟨*symbol*⟩][⟨*offset*⟩] works like the *numbered* \item, but
__enumext_item_star_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.

```
3109  \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
3110    {
3111      \tl_if_novalue:nTF {#1}
3112        {
3113          \tl_gset_eq:Nc
3114            \g__enumext_item_symbol_aux_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
3115        }
3116        {
3117          \tl_gset:Nn \g__enumext_item_symbol_aux_tl {#1}
3118        }
3119      \tl_if_novalue:nTF {#2}
3120        {
3121          \dim_set_eq:cc
3122            { l__enumext_item_symbol_sep_ \__enumext_level: _dim }
3123            { l__enumext_labelsep_ \__enumext_level: _dim }
3124        }
3125        {
3126          \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
3127        }
3128      \bool_if:NT \l__enumext_check_answers_bool
3129        {
3130          \int_gincr:N \g__enumext_item_number_int
3131          \bool_set_true:N \l__enumext_item_number_bool
```

```
3132            }
3133        \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
3134        \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3135      }
```

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

```
3136  \cs_new_protected:Nn \__enumext_item_star_exec:
3137    {
3138      \tl_if_empty:cF { l__enumext_item_symbol_ \__enumext_level: _tl }
3139        {
3140          \mode_leave_vertical:
3141          \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3142          \hbox_overlap_left:n { \g__enumext_item_symbol_aux_tl }
3143          \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3144        }
3145    }
```

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

__enumext_redefine_item: The function __enumext_redefine_item: will redefine the \item command in the enumext environment
__enumext_make_label   adding \item*.

```
3146  \cs_new_protected:Nn \__enumext_redefine_item:
3147    {
3148      \RenewDocumentCommand \item { s o o }
3149        {
3150          \bool_if:nTF {##1}
3151            {
3152              \__enumext_starred_item:nn {##2} {##3}
3153            }
3154            { \__enumext_default_item:n {##2} }
3155        }
3156    }
```

The function __enumext_make_label: redefine \makelabel for the keys align, font, wrap-label, wrap-label* and \item* for enumext environment.

```
3157  \cs_new_protected:Nn \__enumext_make_label:
3158    {
3159      \RenewDocumentCommand \makelabel { m }
3160        {
3161          \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
3162          \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3163          \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3164            {
3165              \__enumext_item_star_exec:
3166              \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3167            }
3168            { ##1 }
3169          \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
3170          \tl_gclear:N \g__enumext_item_symbol_aux_tl
3171        }
3172    }
```

(*End of definition for* __enumext_redefine_item: *and* __enumext_make_label:.)

🌱 This functions are passed to __enumext_list_arg_two_X: used in the definition of the enumext environment (§12.38).

12.34   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*
```
3173  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
3174    {
3175      \keys_define:nn { enumext / #1 }
3176        {
3177          item-sym* .tl_set:c  = { l__enumext_item_symbol_#2_tl },
3178          item-sym* .value_required:n = true,
3179          item-sym* .initial:n  = {$\star$},
3180          item-pos* .dim_set:c  = { l__enumext_item_symbol_sep_#2_dim },
3181          item-pos* .value_required:n = true,
3182        }
```

```
3183      }
3184  \clist_map_inline:nn
3185      {
3186        {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
3187      }
3188    { \__enumext_tmp:nn #1 }
```

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

12.35 Handling unknown keys

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

12.35.1 Handling unknown keys for keyans and keyans*

unknown
__enumext_keyans_unknown_keys:n
__enumext_keyans_unknown_keys:nn

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

```
3189  \cs_set_protected:Npn \__enumext_tmp:n #1
3190    {
3191      \keys_define:nn { enumext / #1 }
3192        {
3193          unknown .code:n = { \__enumext_keyans_unknown_keys:n {##1} }
3194        }
3195    }
3196  \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }
```

Internal functions for handling unknown key.

```
3197  \cs_new_protected:Npn \__enumext_keyans_unknown_keys:n #1
3198    {
3199      \exp_args:NV \__enumext_keyans_unknown_keys:nn \l_keys_key_str {#1}
3200    }
3201  \cs_new_protected:Npn \__enumext_keyans_unknown_keys:nn #1#2
3202    {
3203      \tl_if_blank:nTF {#2}
3204        {
3205          \msg_error:nnn { enumext } { keyans-unknown-key } {#1}
3206        }
3207        {
3208          \msg_error:nnnn { enumext } { keyans-unknown-key-value } {#1} {#2}
3209        }
3210    }
```

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

12.35.2 Handling unknown keys for enumext*

unknown
__enumext_starred_unknown_keys:n
__enumext_starred_unknown_keys:nn

Define and set unknown key for enumext* environment.

```
3211  \keys_define:nn { enumext / enumext* }
3212    {
3213      unknown .code:n = { \__enumext_starred_unknown_keys:n {#1} }
3214    }
```

Internal functions for handling unknown key.

```
3215  \cs_new_protected:Npn \__enumext_starred_unknown_keys:n #1
3216    {
3217      \exp_args:NV \__enumext_starred_unknown_keys:nn \l_keys_key_str {#1}
3218    }
3219  \cs_new_protected:Npn \__enumext_starred_unknown_keys:nn #1#2
3220    {
3221      \tl_if_blank:nTF {#2}
3222        {
3223          \msg_error:nnn { enumext } { starred-unknown-key } {#1}
3224        }
3225        {
3226          \msg_error:nnnn { enumext } { starred-unknown-key-value } {#1} {#2}
3227        }
3228    }
```

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

12.35.3 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.

```
3229 \cs_set_protected:Npn \__enumext_tmp:n #1
3230   {
3231     \keys_define:nn { enumext / #1 }
3232       {
3233         unknown .code:n = { \__enumext_standar_unknown_keys:n {##1} }
3234       }
3235   }
3236 \clist_map_inline:nn { level-1,level-2,level-3,level-4 } { \__enumext_tmp:n {#1} }
```

Internal functions for handling unknown key.

```
3237 \cs_new_protected:Npn \__enumext_standar_unknown_keys:n #1
3238   {
3239     \exp_args:NV \__enumext_standar_unknown_keys:nn \l_keys_key_str {#1}
3240   }
3241 \cs_new_protected:Npn \__enumext_standar_unknown_keys:nn #1#2
3242   {
3243     \tl_if_blank:nTF {#2}
3244       {
3245         \msg_error:nnn { enumext } { standar-unknown-key } {#1}
3246       }
3247       {
3248         \msg_error:nnnn { enumext } { standar-unknown-key-value } {#1} {#2}
3249       }
3250   }
```

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

12.36 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.

```
3251 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
3252   {
3253     \tl_if_novalue:nTF { #1 }
3254       {
3255         \bool_set_true:N \l__enumext_wrap_label_v_bool
3256         \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
3257       }
3258       {
3259         \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
3260         \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
3261       }
3262   }
```

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

__enumext_keyans_starred_item:n

The function __enumext_keyans_starred_item:n which will make a temporary copy of the current ⟨label⟩, execute the show-ans or show-pos keys using the function __enumext_keyans_show_left:n and will display the contents of that item using the internal copy __enumext_item_std:w, this is necessary to prevent incrementing the current "*counter*" of the original ⟨label⟩.

```
3263 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
3264   {
3265     \tl_set_eq:NN \l__enumext_store_current_label_tmp_tl \l__enumext_label_v_tl
3266     \__enumext_keyans_show_left:n { #1 }
3267     \bool_set_true:N \l__enumext_wrap_label_v_bool
3268     \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl \__enumext_keyans_show_item
```

Recover the original value of the current ⟨label⟩ and *store* it first in the ⟨prop list⟩ (including the optional argument), run the internal "*label and ref*" system if the save-ref key is active and finally *store* it in the ⟨sequence⟩.

```
3269     \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_store_current_label_tmp_tl
3270     \__enumext_keyans_addto_prop:n { #1 }
3271     \__enumext_keyans_store_ref:
3272     \__enumext_keyans_addto_seq:n { #1 }
3273     \int_gincr:N \g__enumext_check_starred_cmd_int
3274   }
```

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

\item*

__enumext_keyans_redefine_item:

__enumext_keyans_make_label:

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

```
3275  \cs_new_protected:Nn \__enumext_keyans_redefine_item:
3276    {
3277      \RenewDocumentCommand \item { s o }
3278        {
3279          \bool_if:nTF {##1}
3280            {
3281              \peek_remove_spaces:n
3282                {
3283                  \__enumext_keyans_starred_item:n {##2}
3284                }
3285            }
3286            {
3287              \__enumext_keyans_default_item:n {##2}
3288            }
3289        }
3290    }
```

The function __enumext_keyans_make_label: redefine \makelabel for the keys align, font, wrap-label, wrap-label* and \item* for keyans environment.

```
3291  \cs_new_protected:Nn \__enumext_keyans_make_label:
3292    {
3293      \RenewDocumentCommand \makelabel { m }
3294        {
3295          \tl_use:N \l__enumext_label_fill_left_v_tl
3296          \tl_use:N \l__enumext_label_font_style_v_tl
3297          \bool_if:NTF \l__enumext_wrap_label_v_bool
3298            {
3299              \__enumext_wrapper_label_v:n { ##1 }
3300            }
3301            { ##1 }
3302          \tl_use:N \l__enumext_label_fill_right_v_tl
3303        }
3304    }
```

(*End of definition for* \item*, __enumext_keyans_redefine_item:, *and* __enumext_keyans_make_label:. *This function is documented on page 14.*)

🖌 This functions are passed to __enumext_list_arg_two_v: used in the definition of the keyans environment (§12.37.2).

12.37   Second argument of the lists

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

12.37.1   Calculation of \leftmargin and \itemindent

Consider the figure 9 where the default margins (on the left) of a list are represented.
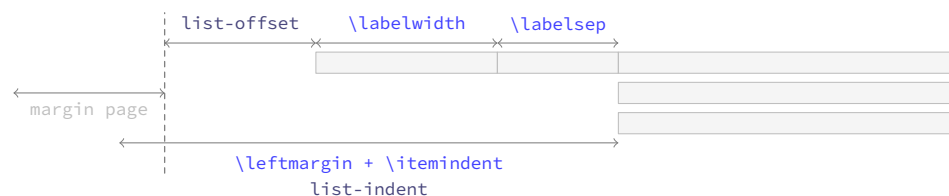
Figure 9: Representation of standard horizontal lengths in list environment.

The idea is to have control over these margins so that our list does not overlap the left margin of the page. The *key* relationship is that the right edge of the \labelsep equals the right edge of the \itemindent, so that the left edge of the *label box* is at \leftmargin+\itemindent minus \labelwidth+\labelsep. Thus, the handling of the margins by the package will be as shown in the figure 10.
Where the default values will look like in the figure 11.

__enumext_calc_hspace:NNNNNNN

__enumext_calc_hspace:ccccccc

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

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

Figure 11: Default horizontal lengths in enumext.

```
#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.

This function is passed to __enumext_list_arg_two_X: which is used in the definition of the enumext and keyans environments (§12.37.2).

```
3305 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
3306   {
3307     \dim_compare:nNnT { #1 } < { \c_zero_dim }
3308       {
3309         \msg_warning:nnnV { enumext } { width-non-positive }{ labelwidth }{ #1 }
3310         \dim_set:Nn #1 { \dim_abs:n { #1 } }
3311       }
3312     \dim_compare:nNnT { #2 } < { \c_zero_dim }
3313       {
3314         \msg_warning:nnnV { enumext } { width-negative }{ labelsep }{ #2 }
3315         \dim_set:Nn #2 { \dim_abs:n { #2 } }
3316       }
```

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

```
3317       \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2} }
```

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

```
3318       \dim_compare:nNnTF { #4 } < { \c_zero_dim }
3319         {
3320           \dim_set:Nn #6 { #1 + #2 - #4}
3321           \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3322         }
3323         {
3324           \dim_compare:nNnT { #4 } = { #1 + #2 }
3325             { \dim_set:Nn #6 { \c_zero_dim } }
3326           \dim_compare:nNnT { #4 } < { #1 + #2 }
3327             { \dim_set:Nn #6 { #1 + #2 - #4} }
3328           \dim_compare:nNnT { #4 } > { #1 + #2 }
3329             {
3330               \dim_set:Nn #6 { -#1 - #2 + #4}
3331               \dim_set:Nn #6 { #6*-1}
3332             }
3333           \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3334         }
3335   }
3336 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { ccccccc }
```

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

12.37.2   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.

```
3337  \cs_set_protected:Npn \__enumext_tmp:n #1
3338    {
3339      \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3340        {
3341          \__enumext_calc_hspace:ccccccc
3342            { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3343            { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3344            { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3345            { l__enumext_leftmargin_tmp_#1_bool }
3346          \clist_map_inline:nn
3347            { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3348            { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
3349          \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3350            { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
3351          \usecounter { enumX#1 }
3352          \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
3353          \str_if_eq:nnTF {#1} { v }
3354            {
3355              \__enumext_keyans_redefine_item:
3356              \__enumext_keyans_make_label:
3357              \__enumext_keyans_ref:
3358              \__enumext_keyans_fake_item:
3359              \bool_if:cT { l__enumext_show_length_#1_bool }
3360                {
3361                  \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3362                }
3363            }
3364            {
3365              \__enumext_redefine_item:
3366              \__enumext_make_label:
3367              \__enumext_standar_ref:
3368              \__enumext_fake_item:
3369              \bool_if:cT { l__enumext_show_length_#1_bool }
3370                {
3371                  \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \l__enumext_level_i
3372                }
3373            }
3374        }
3375    }
3376  \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }
```

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

__enumext_list_arg_two_vii:
__enumext_list_arg_two_viii:

For the horizontal environments enumext* and keyans* the implementation is similar, but, the value of \partopsep is always 0pt. At this point we will modify the parsep key to make it take the value of the itemsep key and later, in the environment definition, we will modify parindent to make it set the value of lisparindent and parsep to set the value of \parskip locally.

```
3377  \cs_set_protected:Npn \__enumext_tmp:n #1
3378    {
3379      \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3380        {
3381          \bool_set_true:c { l__enumext_leftmargin_tmp_#1_bool }
3382          \dim_zero:c { l__enumext_leftmargin_tmp_#1_dim }
3383          \__enumext_calc_hspace:ccccccc
3384            { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3385            { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3386            { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3387            { l__enumext_leftmargin_tmp_#1_bool }
3388          \clist_map_inline:nn
3389            { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3390            { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
3391          \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3392            { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
3393          \skip_set_eq:Nc \parsep  { l__enumext_itemsep_#1_skip }
3394          \skip_zero:N \partopsep
3395          \usecounter { enumX#1 }
3396          \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
```

```
3397        \__enumext_starred_ref:
3398        \str_if_eq:nnTF {#1} { vii }
3399          {
3400            \__enumext_fake_item_vii:
3401            \bool_if:cT { l__enumext_show_length_vii_bool }
3402              { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3403          }
3404          {
3405            \__enumext_fake_item_viii:
3406            \bool_if:cT { l__enumext_show_length_#1_bool }
3407              { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3408          }
3409        }
3410      }
3411    \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:.*)

12.38 The environment enumext

enumext We create the enumext environment based on list environment by levels.

```
3412  \NewDocumentEnvironment{enumext}{ O{} }
3413    {
3414      \__enumext_safe_exec:
3415      \__enumext_parse_keys:n {#1}
3416      \__enumext_before_list:
3417      \__enumext_start_store_level:
3418      \__enumext_start_list:nn
3419        { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
3420        {
3421          \use:c { __enumext_list_arg_two_ \__enumext_level: : }
3422          \__enumext_before_keys_exec:
3423        }
3424      \__enumext_set_item_width:
3425      \__enumext_after_args_exec:
3426    }
3427    {
3428      \__enumext_stop_list:
3429      \__enumext_stop_store_level:
3430      \__enumext_after_list:
3431    }
```

(*End of definition for enumext. This function is documented on page 4.*)

__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.

```
3432  \cs_new_protected:Nn \__enumext_set_item_width:
3433    {
3434      \dim_set:Nn \itemwidth
3435        {
3436          \linewidth
3437        }
3438      \dim_compare:nT
3439        {
3440          \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim } != \c_zero_dim
3441        }
3442        {
3443          \dim_sub:Nn \itemwidth
3444            {
3445              \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3446            }
3447        }
3448    }
```

(*End of definition for __enumext_set_item_width:.*)

__enumext_safe_exec: The __enumext_safe_exec: function first call the function __enumext_internal_mini_page: to create the environment __enumext_mini_page, then the function __enumext_is_not_nested: which sets \g__enumext_standar_bool to "*true*" if we are not nested within enumext*, 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"*.

```
3449  \cs_new_protected:Nn \__enumext_safe_exec:
3450    {
3451      \__enumext_internal_mini_page:
3452      \__enumext_is_not_nested:
3453      \int_incr:N \l__enumext_level_int
3454      \int_compare:nNnT { \l__enumext_level_int } > { 4 }
3455        { \msg_fatal:nn { enumext } { list-too-deep } }
3456      \bool_set_true:N \l__enumext_standar_bool
3457      \bool_set_false:N \l__enumext_starred_bool
3458      \__enumext_is_on_first_level:
3459    }
```

(*End of definition for* __enumext_safe_exec:.)

__enumext_parse_keys:n   The __enumext_parse_store_keys:n function first we will clear the variable \l__enumext_series_-
str used by the key series and then we check if we are at the *"first level"*, if so we process the ⟨keys⟩ and
then execute the function __enumext_parse_series:n used by the key series and call the function
__enumext_nested_base_line_fix: used by the key base-fix, otherwise we will pass the ⟨keys⟩ to
the inner levels of the environment then we execute the function __enumext_store_active_keys:n and
reprocess the ⟨keys⟩ to pass them to the storage ⟨sequence⟩ if the key save-key is not active.

```
3460  \cs_new_protected:Npn \__enumext_parse_keys:n #1
3461    {
3462      \tl_if_novalue:nF {#1}
3463        {
3464          \str_clear:N \l__enumext_series_str
3465          \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
3466            {
3467              \keys_set:nn { enumext / level-1 } {#1}
3468              \__enumext_parse_series:n {#1}
3469              \__enumext_nested_base_line_fix:
3470            }
3471            {
3472              \exp_args:Ne \keys_set:nn
3473                { enumext / level-\int_use:N \l__enumext_level_int } {#1}
3474            }
3475          \__enumext_store_active_keys:n {#1}
3476        }
3477    }
```

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

__enumext_start_store_level: The __enumext_start_store_level: and __enumext_stop_store_level: functions activate the
__enumext_stop_store_level: level saving mechanism for storage in ⟨sequence⟩ for the command \anskey and the environment anskey*.

```
3478  \cs_new_protected:Nn \__enumext_start_store_level:
3479    {
3480      \bool_lazy_all:nT
3481        {
3482          { \bool_if_p:N \l__enumext_store_active_bool }
3483          { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3484          { \bool_if_p:N \g__enumext_standar_bool }
3485        }
3486        {
3487          \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3488            {
3489              \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3490              \__enumext_store_level_open:
3491            }
3492        }
```

If enumext are nested in enumext* add __enumext_store_level_open: to preserve the stored structure.

```
3493      \bool_lazy_all:nT
3494        {
3495          { \bool_if_p:N \l__enumext_store_active_bool }
3496          { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3497          { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
3498        }
3499        {
3500          \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3501            {
```

```
3502                    \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3503                    \__enumext_store_level_open:
3504                }
3505            }
3506    }
```

Close the stored structure.

```
3507    \cs_new_protected:Nn \__enumext_stop_store_level:
3508    {
3509        \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
3510            {
3511                \__enumext_store_level_close:
3512            }
3513    }
```

(*End of definition for* __enumext_start_store_level: *and* __enumext_stop_store_level:.)

__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.

```
3514    \cs_new_protected:Nn \__enumext_before_list:
3515    {
3516        \__enumext_vspace_above:
3517        \__enumext_before_args_exec:
3518        \__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.

```
3519        \dim_compare:nNnT
3520            { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3521            {
3522                \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
3523                {
3524                    \linewidth
3525                    - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
3526                    - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
3527                }
```

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.

```
3528                \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
3529                \int_gincr:N \g__enumext_minipage_stat_int
3530                \__enumext_minipage_add_space:
3531                \__enumext_mini_page{ \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
3532            }
3533        \__enumext_multicols_start:
3534    }
```

(*End of definition for* __enumext_before_list:.)

__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.

```
3535    \cs_new_protected:Nn \__enumext_multicols_start:
3536    {
3537        \int_compare:nNnT
3538            { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3539            {
3540                \dim_compare:nNnT
3541                    { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3542                    {
3543                        \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
3544                            {
3545                                ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
```

```
3546                      + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
3547                    ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
3548                    - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3549                }
3550            }
3551          \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
3552          \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3553            {
3554              \dim_zero:N \columnseprule
3555            }
```

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.

```
3556          \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3557            {
3558              \skip_zero:N \multicolsep
3559              \__enumext_multi_addvspace:
3560            }
3561          \raggedcolumns
3562          \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
3563        }
3564    }
```

(*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.

```
3565 \cs_new_protected:Nn \__enumext_multicols_stop:
3566    {
3567      \int_compare:nNnT
3568        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3569        {
3570          \end{multicols}
3571          \__enumext_unskip_unkern:
3572          \__enumext_unskip_unkern:
3573          \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
3574        }
3575    }
```

(*End of definition for* __enumext_multicols_stop:.)

__enumext_after_list: The function __enumext_after_list: 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.

```
3576 \cs_new_protected:Nn \__enumext_after_list:
3577    {
3578      \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
3579        {
3580          \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3581            {
3582              \msg_warning:nn { enumext } { missing-miniright }
3583              \miniright
3584            }
3585          \int_gzero:N \g__enumext_minipage_stat_int
3586          \__enumext_unskip_unkern: % remove topsep + [partopsep]
3587          \end__enumext_mini_page
3588        }
3589        {
3590          \__enumext_multicols_stop:
3591        }
```

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.

```
3592      \__enumext_after_stop_list:
3593      \__enumext_check_ans_key_hook:
3594      \__enumext_vspace_below:
3595      \bool_set_false:N \l__enumext_standar_bool
```

```
3596        \__enumext_resume_save_counter:
3597      }
```

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.

```
3598 \__enumext_after_env:nn {enumext} { \__enumext_execute_after_env: }
```

(*End of definition for* __enumext_after_list:.)

12.39   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"*.

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

```
3599 \NewDocumentEnvironment{keyans}{ O{} }
3600   {
3601      \__enumext_keyans_safe_exec:
3602      \__enumext_keyans_parse_keys:n {#1}
3603      \__enumext_before_list_v:
3604      \__enumext_start_list:nn
3605        { \tl_use:N \l__enumext_label_v_tl }
3606        {
3607           \__enumext_list_arg_two_v:
3608           \__enumext_before_keys_exec_v:
3609        }
3610      \__enumext_keyans_set_item_width:
3611      \__enumext_after_args_exec_v:
3612   }
3613   {
3614      \__enumext_check_starred_cmd:n { item }
3615      \__enumext_stop_list:
3616      \__enumext_after_list_v:
3617   }
```

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

__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.

```
3618 \cs_new_protected:Nn \__enumext_keyans_set_item_width:
3619   {
3620      \dim_set:Nn \itemwidth
3621        {
3622           \linewidth
3623        }
3624      \dim_compare:nT
3625        {
3626           \l__enumext_listoffset_v_dim != \c_zero_dim
3627        }
3628        {
3629           \dim_sub:Nn \itemwidth
3630             {
3631                \l__enumext_listoffset_v_dim
3632             }
3633        }
3634   }
```

(*End of definition for* __enumext_keyans_set_item_width:.)

__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.

```
3635 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3636   {
3637      \bool_if:NF \l__enumext_store_active_bool
3638        {
3639           \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
3640        }
3641      \int_incr:N \l__enumext_keyans_level_int
3642      \bool_set_true:N \l__enumext_keyans_env_bool
3643      \__enumext_keyans_name_and_start:
```

```
3644      % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3645      \bool_set_false:N \l__enumext_store_active_bool
3646      \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
3647        {
3648          \msg_error:nn { enumext } { keyans-nested }
3649        }
3650      \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3651        {
3652          \msg_error:nn { enumext } { keyans-wrong-level }
3653        }
3654    }
```

(*End of definition for* __enumext_keyans_safe_exec:.)

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

```
3655 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3656    {
3657      \keys_set:nn { enumext / keyans } {#1}
3658    }
```

(*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_after_list_v:

```
3659 \cs_new_protected:Nn \__enumext_before_list_v:
3660    {
3661      \__enumext_vspace_above_v:
3662      \__enumext_before_args_exec_v:
3663      \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3664        {
3665          \dim_set:Nn \l__enumext_minipage_left_v_dim
3666            {
3667              \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3668            }
3669          \bool_set_true:N \l__enumext_minipage_active_v_bool
3670          \int_gincr:N \g__enumext_minipage_stat_int
3671          \__enumext_keyans_minipage_add_space:
3672          \__enumext_mini_page{ \l__enumext_minipage_left_v_dim }
3673        }
3674      \__enumext_keyans_multicols_start:
3675    }
3676 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3677    {
3678      \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3679        {
3680          \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3681            {
3682              \dim_set:Nn \l__enumext_columns_sep_v_dim
3683                {
3684                  (
3685                    \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3686                  ) / \l__enumext_columns_v_int
3687                  - \l__enumext_listoffset_v_dim
3688                }
3689            }
3690          \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim
3691          \dim_zero:N \columnseprule % no rule here
3692          \bool_if:NF \l__enumext_minipage_active_v_bool
3693            {
3694              \skip_zero:N \multicolsep
3695              \__enumext_keyans_multi_addvspace:
3696            }
3697          \raggedcolumns
3698          \begin{multicols}{ \l__enumext_columns_v_int }
3699        }
3700    }
3701 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3702    {
3703      \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3704        {
3705          \end{multicols}
3706          \__enumext_unskip_unkern:
```

```
3707              \__enumext_unskip_unkern:
3708              \par\addvspace{ \l__enumext_multicols_below_v_skip }
3709          }
3710      }
3711  \cs_new_protected:Nn \__enumext_after_list_v:
3712      {
3713          \bool_if:NTF \l__enumext_minipage_active_v_bool
3714              {
3715                  \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3716                      {
3717                          \msg_warning:nn { enumext } { missing-miniright }
3718                          \miniright
3719                      }
3720                  \int_gzero:N \g__enumext_minipage_stat_int
3721                  \__enumext_unskip_unkern: % remove topsep + [partopsep]
3722                  \end__enumext_mini_page
3723                  \par\addvspace{ \l__enumext_minipage_after_skip }
3724              }
3725              {
3726                  \__enumext_keyans_multicols_stop:
3727              }
3728          \bool_set_false:N \l__enumext_keyans_env_bool
3729          \__enumext_after_stop_list_v:
3730          \__enumext_vspace_below_v:
3731      }
```

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

12.40 Tagging PDF support for non-standart list environments

The LaTeX release 2022-06-01 brings automatic support for `tagPDF` in several aspects, including the standart *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 `tagPDF` 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. 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`.

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

start-list-tags
stop-start-tags
stop-list-tags
__enumext_start_list_tag:n
__enumext_stop_start_list_tag:
__enumext_stop_list_tag:n

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

```
3732  \socket_new:nn {tagsupport/enumext/starred}{ 1 }
3733  \socket_new_plug:nnn {tagsupport/enumext/starred} {start-list-tags}
3734      {
3735          \tag_resume:n {#1}
3736          \tag_struct_begin:n {tag=LI}
3737          \tag_struct_begin:n {tag=Lbl}
3738          \tag_mc_begin:n {tag=Lbl}
3739      }
3740  \socket_new_plug:nnn {tagsupport/enumext/starred} {stop-start-tags}
3741      {
3742          \tag_mc_end:
3743          \tag_struct_end:n {tag=Lbl}
3744          \tag_struct_begin:n {tag=LBody}
3745          \tag_struct_begin:n {tag=text-unit}
3746          \tag_struct_begin:n {tag=text}
3747      }
3748  \socket_new_plug:nnn {tagsupport/enumext/starred} {stop-list-tags}
3749      {
3750          \tag_struct_end:n {tag=text}
3751          \tag_struct_end:n {tag=text-unit}
3752          \tag_struct_end:n {tag=LBody}
3753          \tag_struct_end:n {tag=LI}
3754          \tag_suspend:n {#1}
3755      }
```

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

```
3756  \cs_new_protected_nopar:Npn \__enumext_start_list_tag:n #1
3757      {
3758          \IfDocumentMetadataTF
3759              {
3760                  \socket_assign_plug:nn {tagsupport/enumext/starred} {start-list-tags}
```

```
3761        \socket_use:n {tagsupport/enumext/starred} {#1}
3762      }
3763      {}
3764  }
3765 \cs_new_protected_nopar:Nn \__enumext_stop_start_list_tag:
3766  {
3767    \IfDocumentMetadataTF
3768      {
3769        \socket_assign_plug:nn {tagsupport/enumext/starred} {stop-start-tags}
3770        \socket_use:nn {tagsupport/enumext/starred} { }
3771      }
3772      {}
3773  }
3774 \cs_new_protected_nopar:Npn \__enumext_stop_list_tag:n #1
3775  {
3776    \IfDocumentMetadataTF
3777      {
3778        \socket_assign_plug:nn {tagsupport/enumext/starred} {stop-list-tags}
3779        \socket_use:nn {tagsupport/enumext/starred} {#1}
3780      }
3781      {}
3782  }
```

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

12.40.2 Socket for tagging support in keyanspic

<div>

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:

</div>

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

```
3783 \socket_new:nn {tagsupport/enumext/keyanspic}{ 0 }
3784 \socket_new_plug:nnn {tagsupport/enumext/keyanspic} {start-list-tags}
3785  {
3786    \tag_resume:n {keyanspic}
3787    \tag_struct_begin:n {tag=LI}
3788    \tag_struct_begin:n {tag=Lbl}
3789    \tag_mc_begin:n {tag=Lbl}
3790  }
3791 \socket_new_plug:nnn {tagsupport/enumext/keyanspic} {stop-start-tags}
3792  {
3793    \tag_mc_end:
3794    \tag_struct_end:n {tag=Lbl}
3795    \tag_struct_begin:n {tag=LBody}
3796    \tag_struct_begin:n {tag=text-unit}
3797    \tag_struct_begin:n {tag=text}
3798    \tag_mc_begin:n {tag=text}
3799  }
3800 \socket_new_plug:nnn {tagsupport/enumext/keyanspic} {stop-list-tags}
3801  {
3802    \tag_mc_end:
3803    \tag_struct_end:n {tag=text-unit}
3804    \tag_struct_end:n {tag=text}
3805    \tag_struct_end:n {tag=LBody}
3806    \tag_struct_end:n {tag=LI}
3807    \tag_suspend:n {keyanspic}
3808  }
```

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

```
3809 \cs_new_protected_nopar:Nn \__enumext_anspic_start_list_tag:
3810  {
3811    \IfDocumentMetadataTF
3812      {
3813        \socket_assign_plug:nn {tagsupport/enumext/keyanspic} {start-list-tags}
3814        \socket_use:n {tagsupport/enumext/keyanspic}
3815      }
3816      {}
3817  }
3818 \cs_new_protected_nopar:Nn \__enumext_anspic_stop_start_list_tag:
3819  {
3820    \IfDocumentMetadataTF
3821      {
3822        \socket_assign_plug:nn {tagsupport/enumext/keyanspic} {stop-start-tags}
3823        \socket_use:nn {tagsupport/enumext/keyanspic}
3824      }
3825      {}
```

```
3826        }
3827  \cs_new_protected_nopar:Nn \__enumext_anspic_stop_list_tag:
3828    {
3829      \IfDocumentMetadataTF
3830        {
3831          \socket_assign_plug:nn {tagsupport/enumext/keyanspic} {stop-list-tags}
3832          \socket_use:nn {tagsupport/enumext/keyanspic}
3833        }
3834        {}
3835    }
```

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

12.41 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 and are placed inside minipage environments, with the ⟨*label*⟩ underneath, adjusting widths according to the options passed to the environment.

Again it is necessary to "adjust" the spacing, both vertical and horizontal, to obtain an output like the one shown in the figure 12.
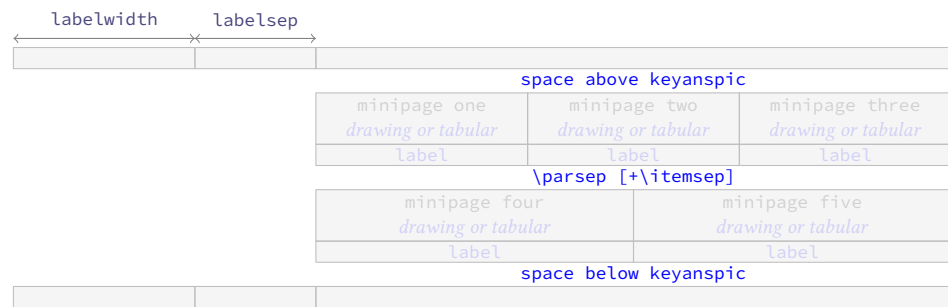
Figure 12: Representation of the keyanspic spacing in enumext.

This implementation is adapted from the answer given by Enrico Gregorio in How to process the body of an environment and divide it by a \macro?.

12.41.1 The environment keyanspic

The environment we wish to build will be based on the list environment and will take two optional arguments, the starred argument * will set the position of the ⟨*label*⟩ to the top if present and bottom otherwise, the second optional argument will process the number of *drawing* or *tabular* that will be on top, bottom or inline when present or not.

__enumext_keyans_pic_safe_exec:n The function __enumext_keyans_pic_safe_exec:n check the starred argument and nested level position inside the enumext environment.

We will set the state of the variable \l__enumext_keyans_pic_star_bool along with the value of the variable \l__enumext_keyans_pic_label_pos_str according to the presence of the starred argument.

```
3836  \cs_new_protected:Npn \__enumext_keyans_pic_safe_exec:n #1
3837    {
3838      \int_incr:N \l__enumext_keyans_pic_level_int
3839      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
3840        {
3841          \msg_error:nn { enumext } { keyanspic-nested }
3842        }
3843      \__enumext_keyans_name_and_start:
3844      \bool_if:nTF { #1 }
3845        {
3846          \bool_set_true:N \l__enumext_keyans_pic_star_bool
3847          \str_set:Nn \l__enumext_keyans_pic_label_pos_str { t }
3848        }
3849        {
3850          \str_set:Nn \l__enumext_keyans_pic_label_pos_str { b }
3851        }
3852    }
```

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

__enumext_keyans_pic_skip_abs:N The function __enumext_keyans_pic_skip_abs:N will return a positive value \parsep.

```
3853  \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3854    {
3855      \dim_compare:nNnT { #1 } < { \c_zero_dim }
```

```
3856          {
3857              \skip_set:Nn #1 { -#1 }
3858          }
3859      }
```

(*End of definition for* __enumext_keyans_pic_skip_abs:N.)

__enumext_keyans_pic_arg_two: The function __enumext_keyans_pic_arg_two: will be used in the second argument of the __enumext_-
start_list:nn function that defines the keyanspic environment, it will handle the setting of spaces.
The first thing to do is to set the boolean variable \l__enumext_leftmargin_tmp_v_bool handled by
the list-indent key to false, then we copy the definition of the second list argument from the keyans
environment.

```
3860  \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
3861      {
3862      \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3863      \__enumext_list_arg_two_v:
```

Ahora guardamos la altura total de nuestra ⟨*labe*⟩ en \l__enumext_anspic_label_htdp_dim

```
3864      \bool_if:NF \l__enumext_keyans_pic_star_bool
3865          {
3866          \hbox_set:Nn \l__enumext_anspic_label_box { \l__enumext_label_v_tl }
3867          \dim_set:Nn \l__enumext_anspic_label_htdp_dim
3868              {
3869              \box_ht_plus_dp:N \l__enumext_anspic_label_box
3870              }
3871          }
```

We will add the value of \itemsep to \parsep which we will use as vertical spacing between the above and
below minipage environments. and adjust the value of \leftmargin, the label and counter are handled
directly by the \anspic command. Then we make equal to zero \labelwidth, \labelsep, \partopsep
and \itemsep so that the horizontal and vertical spacing is not affected.

```
3872      \bool_if:NF \l__enumext_keyans_pic_star_bool
3873          {
3874          \skip_add:Nn \parsep { \l__enumext_anspic_label_htdp_dim }
3875          }
3876      %%%\skip_add:Nn \parsep { \itemsep }
3877      \dim_add:Nn  \leftmargin { -\labelwidth - \labelsep }
3878      \dim_zero:N  \labelwidth
3879      \dim_zero:N  \listparindent
3880      \dim_zero:N  \labelsep
3881      \skip_zero:N \partopsep
3882      \skip_zero:N \itemsep
```

We set the value of \l__enumext_keyans_pic_above_skip which we will use to apply our "adjust" space
above keyanspic, finally we call __enumext_item_std:w followed by \scan_stop: to prevent the error
message returned by LaTeX when not using the \item command.

```
3883      \__enumext_keyans_pic_skip_abs:N \parsep
3884      \skip_set:Nn \l__enumext_keyans_pic_above_skip
3885          {
3886          \box_dp:N \strutbox + \l__enumext_topsep_v_skip %- \parsep
3887          }
3888      }
```

(*End of definition for* __enumext_keyans_pic_arg_two:.)

keyanspic Now we define the environment keyanspic based on list. The optional argument [⟨*number above, number
below*⟩] will determine the number of minipage environments that will be above and below separated by
\parsep+\itemsep within it.

```
3889  \NewDocumentEnvironment{keyanspic}{ s o }
3890      {
3891      \__enumext_keyans_pic_safe_exec:n { #1 }
3892      \begin{list} { } { \__enumext_keyans_pic_arg_two: }
3893      \IfDocumentMetadataTF
3894          {
3895          \tag_suspend:n {list}
3896          }{}
3897      \item[] \scan_stop:
3898      % paranoia
3899      \RenewDocumentCommand \item {}
3900          {
3901          \msg_error:nn { enumext } { keyanspic-item-cmd }
```

```
3902            }
3903        \IfDocumentMetadataTF
3904          {
3905            \tag_resume:n {keyanspic}
3906            \tag_tool:n {para/tagging=false}
3907            \tag_suspend:n {keyanspic}
3908          } { }
3909        %%\par\addvspace{ \l__enumext_keyans_pic_above_skip } % need see what this on space
3910      }
```

If the optional argument is not present, the number of times the \anspic command appears will be counted from \l__enumext_keyans_pic_body_seq and placed in minipage environments on a single line. Finally we check if \anspic* has been used, set the counter to zero and apply our "adjusted" vertical space below the environment.

```
3911      {
3912        \IfDocumentMetadataTF
3913          {
3914            \tag_resume:n {keyanspic}
3915            \tag_struct_begin:n {tag=L,attribute=enumerate}
3916          } { }
3917        \tl_if_novalue:nTF { #2 }
3918          {
3919            \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
3920          }
3921          { \__enumext_keyans_pic_do:n { #2 } }
3922        \IfDocumentMetadataTF
3923          {
3924            \tag_suspend:n {keyanspic}
3925          } { }
3926        \end{list}
3927        \IfDocumentMetadataTF { \tag_struct_end: \tag_struct_end: } { }
3928        \__enumext_check_starred_cmd:n { anspic }
3929        \setcounter { enumXvi } { 0 }
3930        \par\addvspace{ \dim_eval:n { -\parsep + \box_dp:N \strutbox } }
3931        %\bool_set_false:N \l__enumext_store_active_bool
3932      }
```

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

__enumext_keyans_pic_do:n
__enumext_keyans_pic_do:e

The optional argument is split by comma and is handled directly by the function __enumext_keyans_pic_do:n and passed to the function __enumext_keyans_pic_row:n.

```
3933  \cs_new_protected:Nn \__enumext_keyans_pic_do:n
3934    {
3935      \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
3936    }
3937  \cs_generate_variant:Nn \__enumext_keyans_pic_do:n { e }
```

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

__enumext_keyans_pic_row:n

The function __enumext_keyans_pic_row:n will set the widths for the minipage environments and place the content ⟨stored⟩ by \anspic* in the \l__enumext_keyans_pic_body_seq sequence inside them.

```
3938  \cs_new_protected:Nn \__enumext_keyans_pic_row:n
3939    {
3940      \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
3941      \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
3942      \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
3943      \int_step_inline:nnn
3944        { \l__enumext_keyans_pic_above_int + 1 }
3945        { \l__enumext_keyans_pic_below_int }
3946        {
3947          \IfDocumentMetadataTF
3948            {
3949              \tag_suspend:n {minipage}
3950            } { }
3951          \begin{minipage}[ \l__enumext_keyans_pic_label_pos_str ]{ \l__enumext_keyans_pic_width_dir
3952            \centering
3953            \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
3954          \end{minipage}
3955          \IfDocumentMetadataTF
3956            {
3957              \tag_resume:n {minipage}
```

```
3958              } { }
3959          }
3960      \bool_if:NTF \l__enumext_keyans_pic_star_bool
3961          {
3962            \par
3963          }
3964          {
3965            \par%\vspace{ \box_ht_plus_dp:N \strutbox }
3966          }
3967    }
```

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

12.41.2   The command `\anspic`

\anspic   The `\anspic` command take three arguments, the starred (*) versions `\anspic*` and `\anspic*[`⟨*content*⟩`]` store the current ⟨*label*⟩ next to the [⟨*content*⟩] if it is present in the ⟨*sequence*⟩ and ⟨*prop list*⟩ defined by save-ans key. This command is used as a replacement for `\item` in the keyanspic environment.

```
3968  \NewDocumentCommand \anspic { s o +m }
3969    {
```

We check that the command is active in the keyanspic environment only if the save-ans key is present, otherwise we return an error.

```
3970      \bool_if:NF \l__enumext_store_active_bool
3971          {
3972            \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
3973          }
3974      \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3975          {
3976            \msg_error:nn { enumext } { keyanspic-wrong-level }
3977          }
3978      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
3979          {
3980            \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
3981          }
```

The three arguments are handled by the function `\__enumext_keyans_anspic_code:nnn` and stored in the sequence `\l__enumext_keyans_pic_body_seq` which is processed by the keyanspic environment.

```
3982      \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
3983          {
3984            \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
3985          }
3986    }
```

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

__enumext_anspic_box_set_dim:n   The function `\__enumext_keyans_anspic_code:nnn` will be in charge of handling the *"counter"* and ⟨*label*⟩,
__enumext_keyans_anspic_label:nnn   which will have the same configuration as the keyans environment.
__enumext_keyans_anspic_code:nnn

```
3987  \cs_new_protected:Npn \__enumext_anspic_box_set_dim:n #1
3988    {
3989      \vbox_set:Nn \l__enumext_anspic_body_box { #1 }
3990      \dim_set:Nn \l__enumext_anspic_body_htdp_dim
3991          {
3992            \box_ht_plus_dp:N \l__enumext_anspic_body_box
3993          }
3994      %%\vbox_set:Nn \l__enumext_anspic_label_box { \l__enumext_label_v_tl }
3995      %%\dim_set:Nn \l__enumext_anspic_label_htdp_dim
3996      %%  {
3997      %%    \box_ht_plus_dp:N \l__enumext_anspic_label_box
3998      %%  }
3999    }
4000  % process label
4001  \cs_new_protected:Npn \__enumext_anspic_label:nn #1 #2
4002    {
4003      \bool_if:nT { #1 }
4004          {
4005            \__enumext_keyans_addto_prop:n { #2 }
4006            \__enumext_keyans_store_ref:
4007            \__enumext_keyans_addto_seq:n { #2 }
4008            \int_gincr:N \g__enumext_check_starred_cmd_int
4009            \bool_lazy_or:nnT
4010              { \bool_if_p:N \l__enumext_show_answer_bool }
```

```
4011          { \bool_if_p:N \l__enumext_show_position_bool }
4012          {
4013            \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
4014            \__enumext_keyans_show_left:n { #2 }
4015            \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
4016          }
4017        }
4018      \makebox[ \l__enumext_keyans_pic_width_dim ][ c ]
4019        {
4020          \tl_use:N \l__enumext_label_font_style_v_tl
4021          \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
4022        }
4023    }
4024 \cs_new_protected:Npn \__enumext_keyans_anspic_label:nnn #1 #2 #3
4025    {
4026      \stepcounter { enumXvi }
4027      \__enumext_anspic_box_set_dim:n { #3 }
4028      \bool_if:NTF \l__enumext_keyans_pic_star_bool
4029        {
4030          \__enumext_anspic_label:nn { #1 } { #2 }
4031        }
4032        {
4033          \raisebox
4034            {
4035              -\dim_eval:n
4036                {
4037                  \l__enumext_anspic_label_htdp_dim
4038                  + \l__enumext_anspic_body_htdp_dim
4039                  + \box_ht_plus_dp:N \strutbox
4040                }
4041            }
4042            [ 0pt ] [ 0pt ]
4043            {
4044              \__enumext_anspic_label:nn { #1 } { #2 }
4045            }
4046        }
4047    }
4048 \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
4049    {
4050      \__enumext_anspic_start_list_tag:
4051      \__enumext_keyans_anspic_label:nnn { #1 } { #2 } { #3 }
4052      \__enumext_anspic_stop_start_list_tag:
4053      \\ #3
4054      \__enumext_anspic_stop_list_tag:
4055    }
```

(*End of definition for* `\__enumext_anspic_box_set_dim:n`, `\__enumext_keyans_anspic_label:nnn`, *and* `\__enumext_keyans_-anspic_code:nnn`.)

12.42　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` as in the non starred versions, we have no other option than to define a cascade of functions.

12.42.1　Redefining `\footnote` command

`\__enumext_footnotetext:nn`
`\__enumext_renew_footnote:`
`\__enumext_print_footnote:`

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

```
4056 \cs_new_protected:Nn \__enumext_footnotetext:nn
4057    {
4058      \footnotetext[#1]{#2}
4059    }
4060 \cs_new_protected:Nn \__enumext_renew_footnote:
4061    {
4062      \seq_gclear:N \g__enumext_footnote_arg_seq
4063      \seq_gclear:N \g__enumext_footnote_int_seq
4064      \RenewDocumentCommand \footnote { o +m }
4065        {
4066          \tl_if_novalue:nTF {##1}
4067            {
```

```
4068                    \stepcounter{footnote}
4069                    \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
4070                }
4071                {
4072                    \int_gset:Nn \g__enumext_footnote_int { ##1 }
4073                }
4074            \footnotemark [ \g__enumext_footnote_int ]
4075            \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
4076            \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
4077        }
4078    }
4079 \cs_new_protected:Nn \__enumext_print_footnote:
4080    {
4081        \seq_if_empty:NF \g__enumext_footnote_int_seq
4082            {
4083                \seq_map_pairwise_function:NNN
4084                    \g__enumext_footnote_int_seq
4085                    \g__enumext_footnote_arg_seq
4086                    \__enumext_footnotetext:nn
4087            }
4088    }
```

(*End of definition for* __enumext_footnotetext:nn, __enumext_renew_footnote:, *and* __enumext_print_footnote:.)

12.42.2 Functions for item box width

To achieve the horizontal list environment 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 optional first argument (⟨*number*⟩).

__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.

```
4089 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
4090    {
4091        \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
4092            {
4093                \dim_set:Nn \l__enumext_columns_sep_vii_dim
4094                    {
4095                        ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
4096                        / \l__enumext_columns_vii_int
4097                    }
4098            }
4099        \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - 1 }
4100        \dim_set:Nn \l__enumext_item_width_vii_dim
4101            {
4102                ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
4103                / \l__enumext_columns_vii_int
4104                - \l__enumext_labelwidth_vii_dim
4105                - \l__enumext_labelsep_vii_dim
4106            }
```

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

```
4107        \dim_compare:nNnT { \l__enumext_rightmargin_vii_dim } > { \c_zero_dim }
4108            {
4109                \dim_sub:Nn \l__enumext_item_width_vii_dim
4110                    {
4111                        ( \l__enumext_rightmargin_vii_dim * \l__enumext_tmpa_vii_int )
4112                        / \l__enumext_columns_vii_int
4113                    }
4114                \dim_add:Nn \l__enumext_columns_sep_vii_dim
4115                    {
4116                        \l__enumext_rightmargin_vii_dim
4117                    }
4118            }
4119    }
```

Same implementation for the keyans* environment.

```
4120 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
4121    {
4122        \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
4123            {
4124                \dim_set:Nn \l__enumext_columns_sep_viii_dim
```

```
4125          {
4126            ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
4127            / \l__enumext_columns_viii_int
4128          }
4129        }
4130      \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - 1 }
4131      \dim_set:Nn \l__enumext_item_width_viii_dim
4132        {
4133          ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
4134          / \l__enumext_columns_viii_int
4135          - \l__enumext_labelwidth_viii_dim
4136          - \l__enumext_labelsep_viii_dim
4137        }
4138      \dim_compare:nNnT { \l__enumext_rightmargin_viii_dim } > { \c_zero_dim }
4139        {
4140          \dim_sub:Nn \l__enumext_item_width_viii_dim
4141            {
4142              ( \l__enumext_rightmargin_viii_dim * \l__enumext_tmpa_vii_int )
4143              / \l__enumext_columns_viii_int
4144            }
4145          \dim_add:Nn \l__enumext_columns_sep_viii_dim
4146            {
4147              \l__enumext_rightmargin_viii_dim
4148            }
4149        }
4150    }
```

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

12.42.3 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.

```
4151  \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
4152    {
4153      \int_set:Nn \l__enumext_joined_item_vii_int {#1}
4154      \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
4155        {
4156          \msg_warning:nnee { enumext } { item-joined }
4157            { \int_use:N \l__enumext_joined_item_vii_int }
4158            { \int_use:N \l__enumext_columns_vii_int }
4159          \int_set:Nn \l__enumext_joined_item_vii_int
4160            {
4161              \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4162            }
4163        }
4164      \int_compare:nNnT
4165        { \l__enumext_joined_item_vii_int }
4166        >
4167        { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4168        {
4169          \msg_warning:nnee { enumext } { item-joined-columns }
4170            { \int_use:N \l__enumext_joined_item_vii_int }
4171            {
4172              \int_eval:n
4173                { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4174            }
4175          \int_set:Nn \l__enumext_joined_item_vii_int
4176            {
4177              \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4178            }
4179        }
4180      \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { 1 }
4181        {
4182          \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
4183          \int_decr:N \l__enumext_joined_item_aux_vii_int
4184          \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
4185          \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
4186          \dim_set:Nn \l__enumext_joined_width_vii_dim
4187            {
4188              \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
```

```
4189              + (  \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
4190                  + \l__enumext_columns_sep_vii_dim
4191                )*\l__enumext_joined_item_aux_vii_int
4192            }
4193        \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
4194      }
4195      {
4196        \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
4197        \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
4198      }
4199    }
```

Same implementation for the keyans* environment.

```
4200  \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
4201    {
4202      \int_set:Nn \l__enumext_joined_item_viii_int {#1}
4203      \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
4204        {
4205          \msg_warning:nnee { enumext } { item-joined }
4206            { \int_use:N \l__enumext_joined_item_viii_int }
4207            { \int_use:N \l__enumext_columns_viii_int }
4208          \int_set:Nn \l__enumext_joined_item_viii_int
4209            {
4210              \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4211            }
4212        }
4213      \int_compare:nNnT
4214        { \l__enumext_joined_item_viii_int }
4215          >
4216        { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4217        {
4218          \msg_warning:nnee { enumext } { item-joined-columns }
4219            { \int_use:N \l__enumext_joined_item_viii_int }
4220            {
4221              \int_eval:n
4222                { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4223            }
4224          \int_set:Nn \l__enumext_joined_item_viii_int
4225            {
4226              \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4227            }
4228        }
4229      \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { 1 }
4230        {
4231          \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
4232          \int_decr:N \l__enumext_joined_item_aux_viii_int
4233          \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
4234          \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
4235          \dim_set:Nn \l__enumext_joined_width_viii_dim
4236            {
4237              \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
4238              + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
4239                  + \l__enumext_columns_sep_viii_dim
4240                )*\l__enumext_joined_item_aux_viii_int
4241            }
4242          \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
4243        }
4244        {
4245          \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
4246          \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
4247        }
4248    }
```

(*End of definition for* __enumext_starred_joined_item_vii:n *and* __enumext_starred_joined_item_viii:n.)

12.42.4   Functions for mini-env, mini-right and mini-right* keys

__enumext_start_mini_vii: The implementation of the mini-env key support is almost identical to the one used in the enumext and
__enumext_stop_mini_vii: 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.

```
4249  \cs_new_protected:Nn \__enumext_start_mini_vii:
```

```
4250    {
4251      \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
4252        {
4253          \dim_set:Nn \l__enumext_minipage_left_vii_dim
4254            {
4255              \linewidth
4256              - \l__enumext_minipage_right_vii_dim
4257              - \l__enumext_minipage_hsep_vii_dim
4258            }
4259          \bool_set_true:N \l__enumext_minipage_active_vii_bool
4260          \dim_gset_eq:NN
4261            \g__enumext_minipage_right_vii_dim
4262            \l__enumext_minipage_right_vii_dim
4263          \__enumext_mini_addvspace_vii:
4264          \nointerlineskip\noindent
4265          \__enumext_mini_page{ \l__enumext_minipage_left_vii_dim }
4266        }
4267    }
```

The function `\__enumext_stop_mini_vii:` closes the `__enumext_mini_page` environment on the left side, applies `\hfill` and sets the value of the variable `\g__enumext_minipage_active_vii_bool` to true which will be used in the function `\__enumext_after_env:nn` to execute the `__enumext_mini_page` on the *"right side"*.

```
4268  \cs_new_protected:Nn \__enumext_stop_mini_vii:
4269    {
4270      \bool_if:NT \l__enumext_minipage_active_vii_bool
4271        {
4272          \end__enumext_mini_page
4273          \hfill
4274          \bool_gset_true:N \g__enumext_minipage_active_vii_bool
4275        }
4276    }
```

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 `__enumext_mini_page` 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.

```
4277  \__enumext_after_env:nn {enumext*}
4278    {
4279      \bool_if:NT \g__enumext_minipage_active_vii_bool
4280        {
4281          \__enumext_mini_page{ \g__enumext_minipage_right_vii_dim }
4282            \par\addvspace { \g__enumext_minipage_right_skip }
4283            \bool_if:NF \g__enumext_minipage_center_vii_bool
4284              {
4285                \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
4286                  {
4287                    \centering
4288                  }
4289              }
4290            \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
4291              {
4292                \tl_use:N \g__enumext_miniright_code_vii_tl
4293              }
4294            \box_use_drop:N \l__enumext_miniright_code_vii_box
4295          \end__enumext_mini_page
4296          \par\addvspace{ \g__enumext_minipage_after_skip }
4297        }
4298      \bool_gset_false:N \g__enumext_minipage_active_vii_bool
4299      \bool_gset_true:N \g__enumext_minipage_center_vii_bool
4300      \tl_gclear:N \g__enumext_miniright_code_vii_tl
4301      \dim_gzero:N \g__enumext_minipage_right_vii_dim
4302      \bool_gset_false:N \g__enumext_starred_bool
4303    }
```

(*End of definition for* `\__enumext_start_mini_vii:` *and* `\__enumext_stop_mini_vii:`.)

`\__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.

```
4304  \cs_new_protected:Nn \__enumext_start_mini_viii:
4305    {
```

```
4306        \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
4307          {
4308            \dim_set:Nn \l__enumext_minipage_left_viii_dim
4309              {
4310                \linewidth
4311                - \l__enumext_minipage_right_viii_dim
4312                - \l__enumext_minipage_hsep_viii_dim
4313              }
4314            \bool_set_true:N \l__enumext_minipage_active_viii_bool
4315            \dim_gset_eq:NN
4316              \g__enumext_minipage_right_viii_dim
4317              \l__enumext_minipage_right_viii_dim
4318            \__enumext_mini_addvspace_viii:
4319            \nointerlineskip\noindent
4320            \__enumext_mini_page{ \l__enumext_minipage_left_viii_dim }
4321          }
4322      }
4323  \cs_new_protected:Nn \__enumext_stop_mini_viii:
4324    {
4325      \bool_if:NT \l__enumext_minipage_active_viii_bool
4326        {
4327          \end__enumext_mini_page
4328          \hfill
4329          \bool_gset_true:N \g__enumext_minipage_active_viii_bool
4330        }
4331    }
4332  \__enumext_after_env:nn {keyans*}
4333    {
4334      \bool_if:NT \g__enumext_minipage_active_viii_bool
4335        {
4336          \__enumext_mini_page{ \g__enumext_minipage_right_viii_dim }
4337            \par\addvspace { \g__enumext_minipage_right_skip }
4338            \bool_if:NF \g__enumext_minipage_center_viii_bool
4339              {
4340                \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
4341                  {
4342                    \centering
4343                  }
4344              }
4345            \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
4346              {
4347                \tl_use:N \g__enumext_miniright_code_viii_tl
4348              }
4349            \box_use_drop:N \l__enumext_miniright_code_viii_box
4350          \end__enumext_mini_page
4351          \par\addvspace{ \g__enumext_minipage_after_skip }
4352        }
4353      \bool_gset_false:N \g__enumext_minipage_active_viii_bool
4354      \bool_gset_true:N \g__enumext_minipage_center_viii_bool
4355      \tl_gclear:N \g__enumext_miniright_code_viii_tl
4356      \dim_gzero:N \g__enumext_minipage_right_viii_dim
4357    }
```

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

12.43   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.

```
4358  \NewDocumentEnvironment{enumext*}{ o }
4359    {
4360      \__enumext_safe_exec_vii:
4361      \__enumext_parse_keys_vii:n {#1}
4362      \__enumext_before_list_vii:
4363      \__enumext_start_store_level_vii:
4364      \__enumext_start_list:nn { }
4365        {
```

```
4366              \__enumext_list_arg_two_vii:
4367              \__enumext_before_keys_exec_vii:
4368          }
4369        % Stop tagging
4370        \SuspendTagging{enumext*}
4371        \__enumext_starred_columns_set_vii:
4372        \item[] \scan_stop:
4373        \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_first_item_tmp_vii:
4374        \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
4375        \ignorespaces
4376      }
4377      {
4378        % Close for first \item
4379        \IfDocumentMetadataTF { \tag_struct_end: } { }
4380        \__enumext_stop_item_tmp_vii:
4381        \__enumext_remove_extra_parsep_vii:
4382        \__enumext_stop_list:
4383        \__enumext_stop_store_level_vii:
4384        \__enumext_after_list_vii:
4385      }
```

(*End of definition for* enumext*. *This function is documented on page 4.*)

__enumext_safe_exec_vii: We will first call the function __enumext_internal_mini_page: to create the environment __enumext_-mini_page, then the function __enumext_is_not_nested: which sets \g__enumext_starred_bool to true if we are not nested within enumext, 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.

```
4386  \cs_new_protected:Nn \__enumext_safe_exec_vii:
4387    {
4388      \__enumext_internal_mini_page:
4389      \__enumext_is_not_nested:
4390      \int_incr:N \l__enumext_level_h_int
4391      \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
4392        {
4393          \msg_error:nn { enumext } { nested }
4394        }
4395      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
4396        {
4397          \msg_error:nnn { enumext } { nested-horizontal } { keyans*}
4398        }
4399      \bool_set_true:N \l__enumext_starred_bool
4400      \bool_set_false:N \l__enumext_standar_bool
4401      \__enumext_is_on_first_level:
4402    }
```

(*End of definition for* __enumext_safe_exec_vii:.)

__enumext_parse_keys_vii:n First we will clear the variable \l__enumext_series_str used by the key series, process the environment [⟨key = val⟩] and execute the function __enumext_parse_series:n and used by the key series, then we execute the function __enumext_store_active_keys_vii:n and reprocess the ⟨keys⟩ to pass them to the storage ⟨sequence⟩ if the key save-key is not active and finally we call the function __enumext_-nested_base_line_fix: used by the key base-fix.

```
4403  \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
4404    {
4405      \tl_if_novalue:nF {#1}
4406        {
4407          \str_clear:N \l__enumext_series_str
4408          \keys_set:nn { enumext / enumext* } {#1}
4409          \__enumext_parse_series:n {#1}
4410          \__enumext_store_active_keys_vii:n {#1}
4411          \__enumext_nested_base_line_fix:
4412        }
4413    }
```

(*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*.

```
4414 \cs_new_protected:Nn \__enumext_before_list_vii:
4415   {
4416     \__enumext_vspace_above_vii:
4417     \__enumext_check_ans_active:
4418     \__enumext_before_args_exec_vii:
4419     \__enumext_start_mini_vii:
4420   }
```

(*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:` 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.

```
4421 \cs_new_protected:Nn \__enumext_after_list_vii:
4422   {
4423     \__enumext_stop_mini_vii:
4424     \__enumext_after_stop_list_vii:
4425     \__enumext_check_ans_key_hook:
4426     \__enumext_vspace_below_vii:
4427     \bool_set_false:N \l__enumext_starred_bool
4428     \__enumext_resume_save_counter:
4429   }
```

(*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 acti-
`\__enumext_stop_store_level_vii:` vate the level saving mechanism for storage in ⟨*sequence*⟩ of the `\anskey` command and anskey* environment if enumext* are nested in enumext.

```
4430 \cs_new_protected:Nn \__enumext_start_store_level_vii:
4431   {
4432     \bool_if:NT \l__enumext_store_active_bool
4433       {
4434         \int_compare:nNnT { \l__enumext_level_int } > { 0 }
4435           {
4436             \__enumext_store_level_open_vii:
4437           }
4438       }
4439   }
4440 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
4441   {
4442     \bool_if:NT \l__enumext_store_active_bool
4443       {
4444         \int_compare:nNnT { \l__enumext_level_int } > { 0 }
4445           {
4446             \__enumext_store_level_close_vii:
4447           }
4448       }
4449   }
```

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

12.43.1 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.

```
4450 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_vii:
4451   {
4452     \skip_horizontal:n { -\l__enumext_labelwidth_vii_dim - \l__enumext_labelsep_vii_dim }
4453   }
```

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

__enumext_start_item_tmp_vii:

First we will call the function `\__enumext_stop_item_tmp_vii:` that we will redefine later, we will increment the value of `\l__enumext_item_column_pos_vii_int` that will count the item's by rows and the value of `\g__enumext_item_count_all_vii_int` that will count the total of item's in the environment. After that we will call the function `\__enumext_item_peek_args_vii:` that will handle the arguments passed to `\item`.

```
4454  \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
4455    {
4456      \__enumext_stop_item_tmp_vii:
4457      \int_incr:N \l__enumext_item_column_pos_vii_int
4458      \int_gincr:N \g__enumext_item_count_all_vii_int
4459      \__enumext_item_peek_args_vii:
4460    }
```

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

__enumext_item_peek_args_vii:

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).

```
4461  \cs_new_protected:Nn \__enumext_item_peek_args_vii:
4462    {
4463      \peek_meaning:NTF (
4464        { \__enumext_joined_item_vii:w }
4465        { \__enumext_joined_item_vii:w (1) }
4466    }
```

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

__enumext_joined_item_vii:w

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`.

```
4467  \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
4468    {
4469      \__enumext_starred_joined_item_vii:n {#1}
4470      \peek_meaning_remove:NTF *
4471        { \__enumext_starred_item_vii:w  }
4472        { \__enumext_standar_item_vii:w }
4473    }
```

(*End of definition for* `\__enumext_joined_item_vii:w`.)

__enumext_standar_item_vii:w

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 ]`.

```
4474  \cs_new_protected:Npn \__enumext_standar_item_vii:w
4475    {
4476      \bool_set_false:N \l__enumext_item_starred_vii_bool
4477      \peek_meaning:NTF [
4478        {
4479          \bool_set_eq:NN \l__enumext_wrap_label_vii_bool \l__enumext_wrap_label_opt_vii_bool
4480          \__enumext_start_item_vii:w
4481        }
4482        {
4483          \bool_set_true:N \l__enumext_wrap_label_vii_bool
4484          \legacy_if_set_true:n { @noitemarg }
4485          \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4486        }
4487    }
```

(*End of definition for* `\__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

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⟩].

```
4488 \cs_new_protected:Npn \__enumext_starred_item_vii:w
4489   {
4490     \bool_set_true:N \l__enumext_item_starred_vii_bool
4491     \bool_set_true:N \l__enumext_wrap_label_vii_bool
4492     \peek_meaning:NTF [
4493       { \__enumext_starred_item_vii_aux_i:w }
4494       { \__enumext_starred_item_vii_aux_ii:w }
4495   }
4496 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
4497   {
4498     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
4499     \__enumext_starred_item_vii_aux_ii:w
4500   }
4501 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
4502   {
4503     \peek_meaning:NTF [
4504       { \__enumext_starred_item_vii_aux_iii:w }
4505       {
4506         \dim_set_eq:NN \l__enumext_item_symbol_sep_vii_dim \l__enumext_labelsep_vii_dim
4507         \legacy_if_set_true:n { @noitemarg }
4508         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4509       }
4510   }
4511 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
4512   {
4513     \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
4514     \legacy_if_set_true:n { @noitemarg }
4515     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4516   }
```

(*End of definition for* __enumext_starred_item_vii:w *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 *horizontal space.*

```
4517 \cs_new_protected_nopar:Npn \__enumext_fake_make_label_vii:n #1
4518   {
4519     \legacy_if:nT { @noitemarg }
4520       {
4521         \legacy_if_set_false:n { @noitemarg }
4522         \legacy_if:nT { @nmbrlist }
4523           {
4524             \refstepcounter{enumXvii}
4525             \bool_if:NT \l__enumext_check_answers_bool
4526               {
4527                 \int_gincr:N \g__enumext_item_number_int
4528                 \bool_set_true:N \l__enumext_item_number_bool
4529               }
4530           }
4531       }
4532     \bool_if:NT \l__enumext_item_starred_vii_bool
4533       {
4534         \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
4535           {
4536             \tl_gset_eq:NN
4537               \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
4538           }
4539         \mode_leave_vertical:
4540         \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
4541         \hbox_overlap_left:n { \g__enumext_item_symbol_aux_vii_tl }
4542         \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
4543         \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
4544       }
4545     \bool_if:NTF \l__enumext_wrap_label_vii_bool
```

```
4546          {
4547            \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
4548              {
4549                \tl_use:N \l__enumext_label_font_style_vii_tl
4550                  \__enumext_wrapper_label_vii:n {#1}
4551              }
4552          }
4553          {
4554            \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
4555              {
4556                \tl_use:N \l__enumext_label_font_style_vii_tl #1
4557              }
4558          }
4559        \skip_horizontal:N \l__enumext_labelsep_vii_dim
4560      }
```

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

12.43.2   Real definition of `\item` in enumext*

The functions `\__enumext_start_item_vii:w` and `\__enumext_stop_item_vii:` executing the true definition of `\item` inside the enumext* environment, unlike the implementation in shortlst we will NOT use an extra group and the plain form of the lrbox environment.

`\__enumext_start_item_vii:w` The first thing we will do is set the value of `\__enumext_stop_item_tmp_vii:` equal to `\__enumext_stop_item_vii:` which we will define later, after that we will start capturing `\item` and its ⟨*contents*⟩ in a *horizontal box* where the width will be `\itemwidth` plus `\labelwidth` plus `\labelsep`.

```
4561  \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
4562    {
4563      \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
4564      \hbox_set_to_wd:Nnw \l__enumext_item_text_vii_box
4565        {
4566          \l__enumext_joined_width_vii_dim
4567          + \l__enumext_labelwidth_vii_dim
4568          + \l__enumext_labelsep_vii_dim
4569        }
```

If `\DocumentMetadata` is not active and the state of the variable `\l__enumext_footnotes_key_bool` is false, we will redefine the `\footnote` command.

```
4570        \IfDocumentMetadataTF { }
4571          {
4572            \bool_if:NF \l__enumext_footnotes_key_bool
4573              {
4574                \__enumext_renew_footnote:
4575              }
4576          }
```

Now we insert our *sockets* for the tagPDF support and print `\item`.

```
4577          \__enumext_start_list_tag:n {enumext*}
4578          \__enumext_fake_make_label_vii:n {#1}
4579          \__enumext_stop_start_list_tag:
```

Finally we open the minipage environment capture the ⟨*item content*⟩ and execute the first key, listparindent key which will be equal to `\parindent`, the parsep key which will be equal to `\parskip` and the itemindent key.

```
4580          \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim  }
4581            \tl_use:N \l__enumext_after_list_args_vii_tl
4582            \dim_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
4583            \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
4584            \tl_use:N \l__enumext_fake_item_indent_vii_tl
4585      }
```

(*End of definition for* `\__enumext_start_item_vii:w`.)

`\__enumext_stop_item_vii:` The `\__enumext_stop_item_vii:` function will finish the fetching `\item` and its ⟨*content*⟩ by closing the minipage environment, the *sockets* for the tagPDF and the *horizontal box*.

```
4586  \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
4587    {
4588        \__enumext_endminipage:
4589      \__enumext_stop_list_tag:n {enumext*}
4590      \hbox_set_end:
```

Here we will reduce the *warnings* a bit by setting the value of \hbadness to 10000, print the ⟨*contents*⟩ of the *box* along with \footnote.

```
4591        \int_set:Nn \hbadness { 10000 }
4592        \box_use_drop:N \l__enumext_item_text_vii_box
4593        \IfDocumentMetadataTF { }
4594          {
4595            \bool_if:NF \l__enumext_footnotes_key_bool
4596              {
4597                \__enumext_print_footnote:
4598              }
4599          }
```

Finally set the *vertical* and *horizontal* spaces between rows and columns.

```
4600        \int_compare:nNnTF
4601          { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
4602          {
4603            \par\noindent
4604            \int_zero:N \l__enumext_item_column_pos_vii_int
4605          }
4606          {
4607            \skip_horizontal:N \l__enumext_columns_sep_vii_dim
4608          }
4609      }
```

(*End of definition for* __enumext_stop_item_vii:.)

__enumext_remove_extra_parsep_vii: Finally we will remove the vertical space equal to \parsep=\itemsep when the total number of items is divisible by the number of items 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*⟩.

```
4610  \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
4611    {
4612      \int_compare:nNnT
4613        {
4614          \int_mod:nn
4615            {  \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
4616        }
4617        =
4618        { 0 }
4619        {
4620          \para_end:
4621          \skip_vertical:n { -\l__enumext_itemsep_vii_skip }
4622          \skip_vertical:N \c_zero_skip
4623          \int_gzero:N \g__enumext_item_count_all_vii_int
4624        }
4625    }
```

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.

```
4626  \__enumext_after_env:nn {enumext*} { \__enumext_execute_after_env: }
```

(*End of definition for* __enumext_remove_extra_parsep_vii:.)

12.44   The environment keyans*

keyans* First we will generate the environment and we will give a temporary definition to __enumext_stop_item_-tmp_viii: equal to __enumext_first_item_tmp_viii: and next to \item equal to __enumext_-start_item_tmp_viii: which we will redefine later. The implementation of this environment is the same as that used by the enumext* environment except for the __enumext_check_starred_cmd:n function added in the second part.

```
4627  \NewDocumentEnvironment{keyans*}{ o }
4628    {
4629      \__enumext_safe_exec_viii:
4630      \__enumext_parse_keys_viii:n {#1}
4631      \__enumext_before_list_viii:
4632      \__enumext_start_list:nn { }
4633        {
4634          \__enumext_list_arg_two_viii:
4635          \__enumext_before_keys_exec_viii:
4636        }
4637      \__enumext_starred_columns_set_viii:
```

```
4638    \item[] \scan_stop:
4639    \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_first_item_tmp_viii:
4640    \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
4641    \ignorespaces
4642    }
4643    {
4644    \__enumext_stop_item_tmp_viii:
4645    \__enumext_remove_extra_parsep_viii:
4646    \__enumext_check_starred_cmd:n { item }
4647    \__enumext_stop_list:
4648    \__enumext_after_list_viii:
4649    }
```

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

__enumext_safe_exec_viii: The __enumext_safe_exec_viii: function will first check if the save-ans key is active and only when this is true the environment will be available, it will increment the value of \l__enumext_keyans_level_h_int and return an error message when we are nesting the environment, then it will call the __enumext_-keyans_name_and_start: function in charge of saving the name of the environment and the line it is running on, then it will check if we are trying to nest keyans* in enumext* returning an error and we will set \l__enumext_starred_bool to true, finally we will check if we are within the appropriate level within the enumext environment.

```
4650    \cs_new_protected:Nn \__enumext_safe_exec_viii:
4651    {
4652    \bool_if:NF \l__enumext_store_active_bool
4653        {
4654        \msg_error:nnnn { enumext } { wrong-place }{ keyans* }{ save-ans }
4655        }
4656    \int_incr:N \l__enumext_keyans_level_h_int
4657    \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
4658        {
4659        \msg_error:nn { enumext } { nested }
4660        }
4661    \__enumext_keyans_name_and_start:
4662    \bool_if:NT \l__enumext_starred_bool
4663        {
4664        \msg_error:nnn { enumext } { nested-horizontal } { enumext* }
4665        }
4666    \bool_set_true:N \l__enumext_starred_bool
4667    % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
4668    \bool_set_false:N \l__enumext_store_active_bool
4669    \int_compare:nNnT { \l__enumext_level_int } > { 1 }
4670        {
4671        \msg_error:nn { enumext } { keyans-wrong-level }
4672        }
4673    }
```

(*End of definition for __enumext_safe_exec_viii:.*)

__enumext_parse_keys_viii:n   Parse [⟨*key* = *val*⟩] for keyans*.

```
4674    \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
4675    {
4676    \tl_if_novalue:nF {#1}
4677        {
4678        \keys_set:nn { enumext / keyans* } {#1}
4679        }
4680    }
```

(*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.

```
4681    \cs_new_protected:Nn \__enumext_before_list_viii:
4682    {
4683    \__enumext_vspace_above_viii:
4684    \__enumext_before_args_exec_viii:
4685    \__enumext_start_mini_viii:
4686    }
```

(*End of definition for __enumext_before_list_viii:.*)

__enumext_after_list_viii:

The function __enumext_after_list: 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.

```
4687 \cs_new_protected:Nn \__enumext_after_list_viii:
4688   {
4689     \__enumext_stop_mini_viii:
4690     \__enumext_after_stop_list_viii:
4691     \__enumext_vspace_below_viii:
4692   }
```

(*End of definition for* __enumext_after_list_viii:.)

12.44.1 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.

```
4693 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_viii:
4694   {
4695     \skip_horizontal:n { -\l__enumext_labelwidth_viii_dim - \l__enumext_labelsep_viii_dim }
4696   }
```

(*End of definition for* __enumext_first_item_tmp_viii:.)

__enumext_start_item_tmp_viii:

First we will call the function __enumext_stop_item_tmp_viii: that we will redefine later, we will increment the value of \l__enumext_item_column_pos_viii_int that will count the item's by rows and the value of \g__enumext_item_count_all_viii_int that will count the total of item's in the environment. After that we will call the function __enumext_item_peek_args_viii: that will handle the arguments passed to \item.

```
4697 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
4698   {
4699     \__enumext_stop_item_tmp_viii:
4700     \int_incr:N \l__enumext_item_column_pos_viii_int
4701     \int_gincr:N \g__enumext_item_count_all_viii_int
4702     \__enumext_item_peek_args_viii:
4703   }
```

(*End of definition for* __enumext_start_item_tmp_viii:.)

__enumext_item_peek_args_viii:

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).

```
4704 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
4705   {
4706     \peek_meaning:NTF (
4707       { \__enumext_joined_item_viii:w }
4708       { \__enumext_joined_item_viii:w (1) }
4709   }
```

(*End of definition for* __enumext_item_peek_args_viii:.)

__enumext_joined_item_viii:w

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.

```
4710 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
4711   {
4712     \__enumext_starred_joined_item_viii:n {#1}
4713     \peek_meaning_remove:NTF *
4714       { \__enumext_starred_item_viii:w  }
4715       { \__enumext_standar_item_viii:w }
4716   }
```

(*End of definition for* __enumext_joined_item_viii:w.)

__enumext_standar_item_viii:w

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`].

```
4717  \cs_new_protected:Npn \__enumext_standar_item_viii:w
4718    {
4719      \bool_set_false:N \l__enumext_item_starred_viii_bool
4720      \peek_meaning:NTF [
4721        {
4722          \bool_set_eq:NN \l__enumext_wrap_label_viii_bool \l__enumext_wrap_label_opt_viii_bool
4723          \__enumext_start_item_viii:w
4724        }
4725        {
4726          \bool_set_true:N \l__enumext_wrap_label_viii_bool
4727          \legacy_if_set_true:n { @noitemarg }
4728          \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4729        }
4730    }
```

(*End of definition for* `\__enumext_standar_item_viii:w`.)

__enumext_starred_item_viii:w
__enumext_starred_item_viii_aux_i:w
__enumext_starred_item_viii_aux_ii:w

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*⟩`]`.

```
4731  \cs_new_protected:Npn \__enumext_starred_item_viii:w
4732    {
4733      \bool_set_true:N \l__enumext_item_starred_viii_bool
4734      \bool_set_true:N \l__enumext_wrap_label_viii_bool
4735      \peek_meaning:NTF [
4736        { \__enumext_starred_item_viii_aux_i:w }
4737        { \__enumext_starred_item_viii_aux_ii:w }
4738    }
```

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`.

```
4739  \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
4740    {
4741      \tl_clear:N \l__enumext_store_current_label_tl
4742      \tl_if_novalue:nF { #1 }
4743        {
4744          \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
4745            {
4746              \tl_put_right:Ne \l__enumext_store_current_label_tl
4747                {
4748                  \l__enumext_store_keyans_item_opt_sep_tl
4749                }
4750              \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
4751            }
4752          \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
4753        }
4754      \__enumext_starred_item_viii_aux_ii:w
4755    }
4756  \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
4757    {
4758      \legacy_if_set_true:n { @noitemarg }
4759      \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4760    }
```

(*End of definition for* `\__enumext_starred_item_viii:w`, `\__enumext_starred_item_viii_aux_i:w`, *and* `\__enumext_starred_-item_viii_aux_ii:w`.)

__enumext_starred_item_exec:

The function `\__enumext_starred_item_exec:` will be in charge of storing the current ⟨*label*⟩ for `\item*` followed by the [⟨*content*⟩] for `\item*[`⟨*content*⟩`]` if present in the ⟨*sequence*⟩ and ⟨*prop list*⟩ set by the save-ans key. In this same function the keys show-ans, show-pos and save-ref are implemented.

```
4761  \cs_new_protected:Nn \__enumext_starred_item_exec:
4762    {
```

```
4763        \tl_put_left:Ne \l__enumext_store_current_label_tl { \l__enumext_label_viii_tl }
4764        \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
4765        \__enumext_keyans_store_ref:
4766        \tl_put_left:Ne \l__enumext_store_current_label_tl { \item }
4767        \__enumext_keyans_addto_seq_link:
4768        \int_gincr:N \g__enumext_check_starred_cmd_int
4769        \bool_if:NT \l__enumext_show_answer_bool
4770          {
4771            \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4772          }
4773        \bool_if:NT \l__enumext_show_position_bool
4774          {
4775            \tl_set:Ne \l__enumext_mark_answer_sym_tl
4776              {
4777                \group_begin:
4778                  \exp_not:N \normalfont
4779                  \exp_not:N \footnotesize [ \int_eval:n
4780                    {
4781                      \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
4782                    }
4783                  ]
4784                \group_end:
4785              }
4786            \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4787          }
4788      }
```

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

12.44.2   Real definition of `\item` in keyans*

`\__enumext_start_item_viii:w`   The implementation at this point is very similar to that of the enumext* environment.

```
4789  \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
4790    {
4791      \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
4792      \legacy_if:nT { @noitemarg }
4793        {
4794          \legacy_if_set_false:n { @noitemarg }
4795          \legacy_if:nT { @nmbrlist }
4796            {
4797              \bool_if:NT \l__enumext_hyperref_bool
4798                {
4799                  \legacy_if_set_true:n { @hyper@item }
4800                }
4801              \refstepcounter{enumXviii}
4802            }
4803        }
```

Here we start capturing `\item` and its ⟨*contents*⟩ in a *horizontal box*.

```
4804      \hbox_set_to_wd:Nnw \l__enumext_item_text_viii_box
4805        {
4806          \l__enumext_joined_width_viii_dim
4807          + \l__enumext_labelwidth_viii_dim
4808          + \l__enumext_labelsep_viii_dim
4809        }
4810      \bool_if:NF \l__enumext_footnotes_key_bool
4811        {
4812          \__enumext_renew_footnote:
4813        }
4814      \bool_if:NT \l__enumext_item_starred_viii_bool
4815        {
4816          \__enumext_starred_item_exec:
4817        }
4818      \bool_if:NTF \l__enumext_wrap_label_viii_bool
4819        {
4820          \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
4821            {
4822              \tl_use:N \l__enumext_label_font_style_viii_tl
4823              \__enumext_wrapper_label_viii:n {#1}
4824            }
4825        }
4826        {
```

```
4827                    \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
4828                      {
4829                        \tl_use:N \l__enumext_label_font_style_viii_tl #1
4830
4831                      }
4832                  }
4833            \skip_horizontal:N \l__enumext_labelsep_viii_dim
4834            \tl_use:N \l__enumext_after_list_args_viii_tl
4835            \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim  }
4836              \dim_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
4837              \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
4838              \bool_if:NT \l__enumext_item_starred_viii_bool
4839                {
4840                  \tl_use:N \l__enumext_fake_item_indent_viii_tl
4841                  \__enumext_keyans_show_item_opt:
4842                  \skip_horizontal:n { -\l__enumext_fake_item_indent_viii_dim - \l__enumext_labelsep_v
4843                }
4844                {
4845                  \tl_use:N \l__enumext_fake_item_indent_viii_tl
4846                }
4847 %%           First atempt here, need stop tag and set counter for enumi, and more...need test
4848 %%           \RenewDocumentCommand \item { o }
4849 %%             {
4850 %%                \tl_if_novalue:nTF {#1}
4851 %%                  {
4852 %%                     \__enumext_item_std:w
4853 %%                  }
4854 %%                  {
4855 %%                     \__enumext_item_std:w [#1]
4856 %%                  }
4857 %%             }
4858       }
```

(*End of definition for* __enumext_start_item_viii:w.)

__enumext_stop_item_viii: The __enumext_stop_item_viii: function will finish the fetching \item and its ⟨content⟩ by closing the minipage environment and the *horizontal box*. Here we will reduce the *warnings* a bit by setting the value of \hbadness to 10000, print the ⟨contents⟩ of the *box* along with \footnote and finally set the vertical and horizontal spaces between rows and columns.

```
4859 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
4860   {
4861        \__enumext_endminipage:
4862     \hbox_set_end:
4863     \int_set:Nn \hbadness { 10000 }
4864     \box_use_drop:N \l__enumext_item_text_viii_box
4865     \bool_if:NF \l__enumext_footnotes_key_bool
4866       {
4867         \__enumext_print_footnote:
4868       }
4869     \int_compare:nNnTF
4870       { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
4871       {
4872         \par\noindent
4873         \int_zero:N \l__enumext_item_column_pos_viii_int
4874       }
4875       {
4876         \skip_horizontal:N \l__enumext_columns_sep_viii_dim
4877       }
4878   }
```

(*End of definition for* __enumext_stop_item_viii:.)

__enumext_remove_extra_parsep_viii: Finally we will remove the *vertical space* equal to \parsep when the total number of items is divisible by the number of items in the last row of the environment.

```
4879 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
4880   {
4881     \int_compare:nNnT
4882       {
4883         \int_mod:nn
4884           { \g__enumext_item_count_all_viii_int }
4885           { \l__enumext_columns_viii_int }
```

```
4886        }
4887      =
4888      { 0 }
4889      {
4890        \para_end:
4891        \skip_vertical:n { -\l__enumext_itemsep_viii_skip }
4892        \skip_vertical:N \c_zero_skip
4893        \int_gzero:N \g__enumext_item_count_all_viii_int
4894      }
4895    }
```

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

12.45 The command \getkeyans

\getkeyans The \getkeyans command takes a mandatory argument of the form {⟨*store name : position*⟩}. Retrieve a "*single*" content stored by \anskey, \anspic* and \item* from ⟨*prop list*⟩ defined by save-ans key.

```
4896  \NewDocumentCommand \getkeyans { m }
4897    {
4898      \exp_args:Ne \__enumext_getkeyans_aux:n
4899        { \tl_to_str:e { \text_expand:n {#1} } }
4900    }
```

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

__enumext_getkeyans_aux:n The internal function __enumext_getkeyans_aux:n is in charge of *splitting* the ⟨*argument*⟩ using ":". If ":" is omitted it will return an error.

```
4901  \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
4902    {
4903      \str_if_in:nnTF {#1} { : }
4904        {
4905          \use:e
4906            {
4907              \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
4908                { {##1} {##2} }
4909            }
4910          \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
4911        }
4912        { \msg_error:nnn { enumext } { missing-colon } {#1} }
4913    }
```

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

__enumext_getkeyans:nn 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*⟩.

```
4914  \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
4915    {
4916      \prop_if_exist:cTF { g__enumext_#1_prop }
4917        {
4918          \prop_item:cn { g__enumext_#1_prop }{#2}
4919        }
4920        {
4921          \msg_error:nnn { enumext } { undefined-storage-anskey } {#1}
4922        }
4923    }
```

(*End of definition for* `\__enumext_getkeyans:nn`.)

12.46 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*⟩].

```
4924  \keys_define:nn  { enumext / print }
```

```
4925        {
4926          print*  .code:n     = \keys_precompile:neN { enumext / enumext* }
4927                                 { \__enumext_filter_save_key:n {#1} }
4928                                 \l__enumext_print_keyans_starred_tl, % starred cmd
4929          print*  .initial:n  = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4930          print-1 .code:n     = \keys_precompile:neN { enumext / level-1 }
4931                                 { \__enumext_filter_save_key:n {#1} }
4932                                 \l__enumext_print_keyans_i_tl,
4933          print-1 .initial:n  = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4934          print-2 .code:n     = \keys_precompile:neN { enumext / level-2 }
4935                                 { \__enumext_filter_save_key:n {#1} }
4936                                 \l__enumext_print_keyans_ii_tl,
4937          print-2 .initial:n  = { nosep, label=(\alph*), first=\small, font=\small },
4938          print-3 .code:n     = \keys_precompile:neN { enumext / level-3 }
4939                                 { \__enumext_filter_save_key:n {#1} }
4940                                 \l__enumext_print_keyans_iii_tl,
4941          print-3 .initial:n  = { nosep, label=\roman*., first=\small, font=\small },
4942          print-4 .code:n     = \keys_precompile:neN { enumext / level-4 }
4943                                 { \__enumext_filter_save_key:n {#1} }
4944                                 \l__enumext_print_keyans_iv_tl,
4945          print-4 .initial:n  = { nosep, label=\Alph*., first=\small, font=\small },
4946          print-* .code:n     = \keys_precompile:neN { enumext / enumext* }
4947                                 { \__enumext_filter_save_key:n {#1} }
4948                                 \l__enumext_print_keyans_vii_tl, % starred nested
4949          print-* .initial:n  = { nosep, label=\arabic*., first=\small, font=\small },
4950        }
```

💣 The reason for storing ⟨keys⟩ in token lists using \keys_precompile:neN is because the keys are set via \setenumext but are later executed by running the command \printkeyans and they are not handled directly by its optional argument, except those related to the first opening level.

\printkeyans   Create a user command to print "all stored content" in ⟨sequence⟩ for \anskey, anskey*, \item* and \anspic*. Within a group we will run our "precompiled keys" and then call the internal function __enumext_printkeyans:nnn.

```
4951  \NewDocumentCommand \printkeyans { s O{} m }
4952    {
4953      \group_begin:
4954        \tl_use:N \l__enumext_print_keyans_i_tl
4955        \tl_use:N \l__enumext_print_keyans_ii_tl
4956        \tl_use:N \l__enumext_print_keyans_iii_tl
4957        \tl_use:N \l__enumext_print_keyans_iv_tl
4958        \tl_use:N \l__enumext_print_keyans_vii_tl
4959        \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
4960      \group_end:
4961    }
```

(*End of definition for* \printkeyans. *This function is documented on page 16.*)

__enumext_printkeyans:nnn   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.

```
4962  \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
4963    {
4964      \seq_if_exist:cTF { g__enumext_#3_seq }
4965        {
4966          \seq_if_empty:cF { g__enumext_#3_seq }
4967            {
4968              %%\seq_show:c { g__enumext_#3_seq }
```

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*, it will open the environment enumext* passing the optional argument to the "first level", set the key base-fix and then will map the ⟨sequence⟩.

```
4969              \bool_if:nTF {#1}
4970                {
4971                  \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
4972                    {
4973                      \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
4974                    }
4975                    {
4976                      \tl_use:N \l__enumext_print_keyans_starred_tl
4977                      \begin{enumext*}[#2]
```

```
4978                    \keys_set:nn { enumext / level-1 }{ base-fix }
4979                    \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4980                  \end{enumext*}
4981                }
4982            }
```

Otherwise it will open the environment enumext passing the optional argument to the *"first level"*, set the key base-fix and then map the ⟨*sequence*⟩.

```
4983            {
4984              \begin{enumext}[#2]
4985                \keys_set:nn { enumext / enumext* }{ base-fix }
4986                \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4987              \end{enumext}
4988            }
4989          }
4990        }
4991        {
4992          \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
4993        }
4994    }
```

(*End of definition for* __enumext_printkeyans:nnn.)

12.47 The command \setenumext

The command \setenumext will be in charge of managing the ⟨*keys*⟩ passed to all environments and to the \printkeyans command. We must take precautions with the enumext* environment and *"first level"* of the enumext environment so as not to capture ⟨*keys*⟩ that complicate us.

__enumext_filter_first_level:n
__enumext_filter_first_level_key:n
__enumext_filter_first_level_pair:nn

The function __enumext_filter_first_level:n will be in charge of filtering the ⟨*keys*⟩ passed to the environment enumext* and *"first level"* of the environment enumext.

```
4995 \cs_new:Npn \__enumext_filter_first_level:n #1
4996    {
4997      \use:e
4998        {
4999          \keyval_parse:NNn
5000            \__enumext_filter_first_level_key:n
5001            \__enumext_filter_first_level_pair:nn {#1}
5002        }
5003    }
```

The function __enumext_filter_first_level_key:n will be responsible for filtering the ⟨*keys*⟩ that are passed *"without value"* by excluding the keys resume and resume*.

```
5004 \cs_new:Npn \__enumext_filter_first_level_key:n #1
5005    {
5006      \str_case:nnF {#1}
5007        {
5008          { resume   } {}
5009          { resume*  } {}
5010        }
5011        { , { \exp_not:n {#1} } }
5012    }
```

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

```
5013 \cs_new:Npn \__enumext_filter_first_level_pair:nn #1#2
5014    {
5015      \str_case:nnF {#1}
5016        {
5017          { series } {}
5018          { resume } {}
5019          { save-ans } {}
5020        }
5021        { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
5022    }
```

(*End of definition for* __enumext_filter_first_level:n, __enumext_filter_first_level_key:n, *and* __enumext_filter_first_level_pair:nn.)

Now define a *"meta families"* of ⟨*keys*⟩ to access from \setenumext.

```
5023 \keys_define:nn { enumext / meta-families }
5024    {
5025      enumext-1 .code:n =
5026              {
```

```
5027              \keys_set:ne { enumext / level-1 }
5028                {
5029                  \__enumext_filter_first_level:n {#1}
5030                }
5031            } ,
5032      enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
5033      enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
5034      enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
5035      keyans    .code:n = { \keys_set:nn { enumext / keyans  } {#1} } ,
5036      enumext*  .code:n =
5037              {
5038                \keys_set:ne { enumext / enumext* }
5039                  {
5040                    \__enumext_filter_first_level:n {#1}
5041                  }
5042              } ,
5043      keyans*   .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
5044      print*    .code:n = { \keys_set:nn { enumext / print   } { print*  = {#1} } } ,
5045      print-1   .code:n = { \keys_set:nn { enumext / print   } { print-1 = {#1} } } ,
5046      print-2   .code:n = { \keys_set:nn { enumext / print   } { print-2 = {#1} } } ,
5047      print-3   .code:n = { \keys_set:nn { enumext / print   } { print-3 = {#1} } } ,
5048      print-4   .code:n = { \keys_set:nn { enumext / print   } { print-4 = {#1} } } ,
5049      print-*   .code:n = { \keys_set:nn { enumext / print   } { print-* = {#1} } } ,
5050      unknown   .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
5051    }
```

We store them in the constant sequence \c__enumext_all_families_seq separated by commas.

```
5052 \seq_const_from_clist:Nn \c__enumext_all_families_seq
5053   {
5054     enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
5055     keyans*, print-1, print-2, print-3, print-4, print-*, print*,
5056   }
```

\setenumext   Now we define the user command \setenumext.

```
5057 \NewDocumentCommand \setenumext { O{enumext,1} +m }
5058   {
5059     \seq_clear:N \l__enumext_setkey_tmpa_seq
5060     \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
5061     \int_set:Nn \l__enumext_setkey_tmpa_int
5062       {
5063         \seq_count:N \l__enumext_setkey_tmpb_seq
5064       }
5065     \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
5066       {
5067         \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
5068         \seq_map_function:NN \l__enumext_setkey_tmpb_seq \__enumext_set_parse:n
5069         \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
5070           {
5071             \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
5072           }
5073       }
5074       {
5075         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
5076       }
5077     \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
5078       { \seq_map_inline:Nn \c__enumext_all_families_seq }
5079       { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
5080       {
5081         \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
5082       }
5083   }
```

(*End of definition for* \setenumext. *This function is documented on page 6.*)

__enumext_set_parse:n   Internal functions used by the \setenumext command.
__enumext_set_error:nn

```
5084 \cs_new_protected:Npn \__enumext_set_parse:n #1
5085   {
5086     \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
5087     \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
5088       { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
5089     \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
```

```
5090        {
5091            \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
5092                { \tl_trim_spaces:n {#1} }
5093        }
5094        { \__enumext_set_error:nn {#1} { } }
5095    }
5096 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
5097    { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }
```

(*End of definition for* __enumext_set_parse:n *and* __enumext_set_error:nn.)

12.48 The command \setenumextmeta

The command \setenumextmeta will be responsible for adding new *"meta-keys"* for the enumext and enumext* environments. The implementation code was given by Jonathan P. Spratte (@Skillmon) answer in Add .meta key to existing keys (l3keys).

\setenumextmeta

\c__enumext_meta_paths_prop

__enumext_add_meta_key:nnn

__enumext_def_meta_key:nnn

__enumext_def_meta_key:Vnn

First we will create a prop list \c__enumext_meta_paths_prop to handle the optional argument.

```
5098 \prop_const_from_keyval:Nn \c__enumext_meta_paths_prop
5099    {
5100        {enumext,1} = level-1,
5101        {enumext,2} = level-2,
5102        {enumext,3} = level-3,
5103        {enumext,4} = level-4,
5104        {enumext*}  = enumext*
5105    }
```

Now we create the user command taking care that unknown cannot be passed as an argument.

```
5106 \NewDocumentCommand \setenumextmeta { s O{enumext,1} m +m }
5107    {
5108        \str_if_eq:eeTF { \tl_trim_spaces:n {#3} } { unknown }
5109            { \msg_error:nn { enumext } { prohibited-unknown } }
5110            {
5111                \bool_if:nTF {#1}
5112                    {
5113                        \int_step_inline:nn { 4 }
5114                            { \__enumext_add_meta_key:nnn { enumext, ##1 } {#3} {#4} }
5115                        \__enumext_add_meta_key:nnn { enumext* } {#3} {#4}
5116                    }
5117                    { \__enumext_add_meta_key:nnn {#2} {#3} {#4} }
5118            }
5119    }
```

The internal functions __enumext_add_meta_key:nnn and __enumext_def_meta_key:nnn will check the optional argument and create the *"meta-key"*.

```
5120 \cs_new_protected:Npn \__enumext_add_meta_key:nnn #1
5121    {
5122        \tl_set:Nn \l__enumext_meta_path_tl {#1}
5123        \tl_replace_all:Nnn \l__enumext_meta_path_tl { ~ } {}
5124        \prop_get:NVNTF
5125            \c__enumext_meta_paths_prop \l__enumext_meta_path_tl \l__enumext_meta_path_tl
5126            { \__enumext_def_meta_key:Vnn \l__enumext_meta_path_tl }
5127            {
5128                \msg_error:nnn { enumext } { unknown-set } {#1}
5129                \use_none:nn
5130            }
5131    }
5132 \cs_new_protected:Npn \__enumext_def_meta_key:nnn #1#2#3
5133    {
5134        \bool_lazy_or:nnTF
5135            { \keys_if_exist_p:nn { enumext / #1 } {#2} }
5136            { \keys_if_exist_p:nn { enumext / enumext* } {#2} }
5137            { \msg_error:nnn { enumext } { already-defined } {#2} }
5138            {
5139                \keys_define:nn { enumext / #1 }
5140                    {
5141                        #2 .meta:n = {#3},
5142                        #2 .value_forbidden:n = true
5143                    }
5144            }
5145    }
5146 \cs_generate_variant:Nn \__enumext_def_meta_key:nnn { V }
```

(*End of definition for* \setenumextmeta *and others. This function is documented on page 6.*)

12.49 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.

<div style="text-align: right">

\foreachkeyans
__enumext_parse_foreach_keys:nn
__enumext_parse_foreach_keys:n
__enumext_foreach_keyans:nn
__enumext_foreach_add_body:n

</div>

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.

```
5147  \keys_define:nn { enumext / foreach }
5148    {
5149      before  .tl_set:N  = \l__enumext_foreach_before_tl,
5150      before  .value_required:n = true,
5151      after   .tl_set:N  = \l__enumext_foreach_after_tl,
5152      after   .value_required:n = true,
5153      start   .int_set:N = \l__enumext_foreach_start_int,
5154      start   .value_required:n = true,
5155      stop    .int_set:N = \l__enumext_foreach_stop_int,
5156      stop    .value_required:n = true,
5157      step    .int_set:N = \l__enumext_foreach_step_int,
5158      step    .value_required:n = true,
5159      wrapper .cs_set_protected:Np = \__enumext_foreach_wrapper:n #1,
5160      wrapper .value_required:n = true,
5161      sep     .tl_set:N  = \l__enumext_foreach_sep_tl,
5162      sep     .value_required:n = true,
5163      unknown .code:n    = { \__enumext_parse_foreach_keys:n {#1} }
5164    }
5165  \keys_precompile:nnN { enumext / foreach }
5166    {
5167      before={},after={},start=1,step=1,stop=0,wrapper=#1,sep=
5168    }
5169    \g__enumext_foreach_default_keys_tl
```

Functions for handling unknown ⟨*keys*⟩.

```
5170  \cs_new_protected:Npn \__enumext_parse_foreach_keys:nn #1#2
5171    {
5172      \tl_if_blank:nTF {#2}
5173        {
5174          \msg_error:nnn { enumext } { for-key-unknown } {#1}
5175        }
5176        {
5177          \msg_error:nnnn { enumext } { for-key-value-unknown } {#1} {#2}
5178        }
5179    }
5180  \cs_new_protected:Npn \__enumext_parse_foreach_keys:n #1
5181    {
5182      \exp_args:NV \__enumext_parse_foreach_keys:nn \l_keys_key_str {#1}
5183    }
```

We create the command.

```
5184  \NewDocumentCommand \foreachkeyans { +O{} m }
5185    {
5186      \__enumext_foreach_keyans:nn {#1} {#2}
5187    }
```

Finally the internal functions __enumext_foreach_keyans:nn and __enumext_foreach_add_body:n will loop through the prop list and print the contents.

```
5188  \cs_new_protected:Npn \__enumext_foreach_keyans:nn #1 #2
5189    {
5190      \tl_use:N \g__enumext_foreach_default_keys_tl
5191      \keys_set:nn { enumext / foreach } {#1}
5192      \tl_set:Nn \l__enumext_foreach_name_prop_tl {#2}
5193      \prop_if_exist:cF { g__enumext_#2_prop }
5194        {
5195          \msg_error:nnn { enumext } { undefined-storage-anskey } {#2}
5196        }
5197      \int_compare:nNnT { \l__enumext_foreach_stop_int } = { 0 }
5198        {
5199          \int_set:Nn \l__enumext_foreach_stop_int
5200            { \prop_count:c { g__enumext_#2_prop } }
5201        }
5202      \seq_clear:N \l__enumext_foreach_print_seq
5203      \int_step_function:nnnN
```

```
5204        { \l__enumext_foreach_start_int }
5205        { \l__enumext_foreach_step_int }
5206        { \l__enumext_foreach_stop_int }
5207      \__enumext_foreach_add_body:n
5208      \seq_use:NV \l__enumext_foreach_print_seq \l__enumext_foreach_sep_tl
5209    }
5210  \cs_new_protected:Npn \__enumext_foreach_add_body:n #1
5211    {
5212      \seq_put_right:Ne \l__enumext_foreach_print_seq
5213        {
5214          \exp_not:V \l__enumext_foreach_before_tl
5215          \__enumext_foreach_wrapper:n
5216            {
5217              \prop_item:cn { g__enumext_ \l__enumext_foreach_name_prop_tl _prop }{#1}
5218            }
5219          \exp_not:V \l__enumext_foreach_after_tl
5220        }
5221    }
```

(*End of definition for* \foreachkeyans *and others. This function is documented on page* *16*.)

12.50   Messages

Message used by package-load for multicol and hyperref packages.

```
5222  \msg_new:nnn { enumext } { package-load }
5223    {
5224      The ~ '#1' ~ package ~ is ~ already ~ loaded.
5225    }
5226  \msg_new:nnn { enumext } { package-not-load }
5227    {
5228      The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
5229    }
5230  \msg_new:nnn { enumext } { package-load-foot }
5231    {
5232      The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
5233    }
```

Message used in the creation of counters by enumext package.

```
5234  \msg_new:nnn { enumext } { counters }
5235    {
5236      The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
5237      package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
5238    }
```

Message used by align and mark-pos keys.

```
5239  \msg_new:nnn { enumext } { unknown-choice }
5240    {
5241      The ~ value ~ '#3' ~ for ~ '#1' ~ key ~ is ~ invalid ~ use ~ ('#2').
5242    }
```

Message used by reserved anskey* environment by enumext package.

```
5243  \msg_new:nnnn { enumext } { anskey-env-error }
5244    {
5245      The ~ '#1' ~ environment ~is ~ reserved ~ by ~\\
5246      'enumext' ~ package, ~ It~ is~ already~ defined.
5247    }
5248    {
5249      The ~ anskey* ~ environment ~ is ~ defined ~ internally ~
5250      for ~ the ~ 'save-ans' ~ key.\\
5251    }
```

Message used in the creation of ⟨*prop list*⟩ by enumext package.

```
5252  \msg_new:nnn { enumext } { store-prop }
5253    {
5254      * ~ Package ~ enumext: ~ Creating ~
5255      \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
5256    }
5257  \msg_new:nnn { enumext } { store-seq }
5258    {
5259      * ~ Package ~ enumext: ~ Creating ~
5260      \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
5261    }
5262  \msg_new:nnn { enumext } { store-int }
```

```
5263        {
5264          * ~ Package ~ enumext: ~ Creating ~
5265          \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_context:.
5266        }
5267   \msg_new:nnn { enumext } { prop-seq-int-hook }
5268        {
5269          * ~ Package ~ enumext: ~ Elements ~ in ~
5270          \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
5271          * ~ Package ~ enumext: ~ Elements ~ in ~
5272          \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
5273          * ~ Package ~ enumext: ~ Value ~ off ~
5274          \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
5275        }
5276   \msg_new:nnn { enumext } { item-answer-hook }
5277        {
5278          * ~ Package ~ enumext: ~ Value ~ off ~
5279          \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\\
5280          * ~ Package ~ enumext: ~ Value ~ off ~
5281          \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\\
5282          * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
5283        }
```

Message used by [⟨*key* = *val*⟩] system and \setenumext command.

```
5284   \msg_new:nnn { enumext } { invalid-key }
5285        {
5286          The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
5287        }
5288   \msg_new:nnn { enumext } { unknown-key-family }
5289        {
5290          Unknown~key~family~`\l_keys_key_str'~for~enumext.
5291        }
```

Messages used in length calculation.

```
5292   \msg_new:nnn { enumext } { width-negative }
5293        {
5294          Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\\
5295          The ~ key ~ '#1'~ accepts ~ values  ~ >= ~ 0pt.
5296        }
5297   \msg_new:nnn { enumext } { width-zero }
5298        {
5299          Invalid ~ '#1=#2' ~ \msg_line_context:.\\
5300          The ~ key ~ '#1'~ accepts ~ values  ~ > ~ 0pt.
5301        }
```

Messages used by show-length key in enumext.

```
5302   \msg_new:nnn { enumext } { list-lengths }
5303        {
5304          **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
5305          \__enumext_show_length:nnn { dim  } { labelsep    } {#1}
5306          \__enumext_show_length:nnn { dim  } { labelwidth  } {#1}
5307          \__enumext_show_length:nnn { dim  } { itemindent  } {#1}
5308          \__enumext_show_length:nnn { dim  } { leftmargin  } {#1}
5309          \__enumext_show_length:nnn { dim  } { rightmargin  } {#1}
5310          \__enumext_show_length:nnn { dim  } { listparindent } {#1}
5311          \__enumext_show_length:nnn { skip } { topsep    } {#1}
5312          \__enumext_show_length:nnn { skip } { parsep    } {#1}
5313          \__enumext_show_length:nnn { skip } { partopsep } {#1}
5314          \__enumext_show_length:nnn { skip } { itemsep   } {#1}
5315          ****************************************************
5316        }
```

Messages used by show-length key in enumext*, keyans* and keyans.

```
5317   \msg_new:nnn { enumext } { list-lengths-not-nested }
5318        {
5319          **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
5320          \__enumext_show_length:nnn { dim  } { labelsep    } {#1}
5321          \__enumext_show_length:nnn { dim  } { labelwidth  } {#1}
5322          \__enumext_show_length:nnn { dim  } { itemindent  } {#1}
5323          \__enumext_show_length:nnn { dim  } { leftmargin  } {#1}
5324          \__enumext_show_length:nnn { dim  } { rightmargin  } {#1}
5325          \__enumext_show_length:nnn { dim  } { listparindent } {#1}
5326          \__enumext_show_length:nnn { skip } { topsep    } {#1}
5327          \__enumext_show_length:nnn { skip } { parsep    } {#1}
```

```
5328        \__enumext_show_length:nnn { skip } { partopsep } {#1}
5329        \__enumext_show_length:nnn { skip } { itemsep   } {#1}
5330        ***********************************************
5331     }
```

Messages used by ref key.

```
5332  \msg_new:nnn { enumext } { key-ref-empty }
5333    {
5334      Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
5335    }
```

Messages used by save-ans key.

```
5336  \msg_new:nnn { enumext } { save-ans-empty }
5337    {
5338      Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
5339    }
5340  \msg_new:nnn { enumext } { save-ans-log }
5341    {
5342      * ~ Package ~ enumext: ~ Start ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5343    }
5344  \msg_new:nnn { enumext } { save-ans-log-hook }
5345    {
5346      * ~ Package ~ enumext: ~ Stop ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5347    }
5348  \msg_new:nnn { enumext } { save-ans-hook }
5349    {
5350      Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
5351    }
```

Messages used by the internal system to check answer used by check-ans key.

```
5352  \msg_new:nnn { enumext } { need-save-ans }
5353    {
5354      Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
5355    }
5356  \msg_new:nnn { enumext } { items-same-answer }
5357    {
5358      ****************************************\\
5359      * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~
5360      for ~ \c_left_brace_str #2 \c_right_brace_str\\
5361      * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~
5362      'OK', ~ all ~ items ~ with ~ answer.\\
5363      ****************************************
5364    }
5365  \msg_new:nnn { enumext } { item-greater-answer }
5366    {
5367      Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
5368      started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
5369      Items ~ > ~ Answers.
5370    }
5371  \msg_new:nnn { enumext } { item-less-answer }
5372    {
5373      Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
5374      started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
5375      Items ~ < ~ Answers.
5376    }
```

Messages used by the internal system to check for *"starred"* \item* and \anspic* commands.

```
5377  \msg_new:nnn { enumext } { missing-starred }
5378    {
5379      Missing ~ '\c_backslash_str #1*' ~ #2.
5380    }
5381  \msg_new:nnn { enumext } { many-starred }
5382    {
5383      Many ~ '\c_backslash_str #1*' ~ #2.
5384    }
```

Messages used by \printkeyans* command.

```
5385  \msg_new:nnn { enumext } { print-starred }
5386    {
5387      \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
5388      #2 ~ environment ~  \msg_line_context:.
5389    }
```

Message for the nesting depth of the environment enumext.

```
5390  \msg_new:nnn { enumext } { list-too-deep }
5391    {
5392      Too ~ deep ~ nesting  ~ for ~ 'enumext' ~ \msg_line_context:.~ \\
5393      The ~ maximum  ~ level  ~ of  ~ nesting  ~ is ~ 4.
5394    }
```

Messages used by \anskey, anskey* and \anspic commands.

```
5395  \msg_new:nnn { enumext } { anskey-unnumber-item }
5396    {
5397      Can't ~ store ~ with ~ a ~ unnumbered ~ \c_backslash_str item ~ \msg_line_context:.
5398    }
5399  \msg_new:nnn { enumext } { anskey-already-stored }
5400    {
5401      Content ~ already ~ stored ~ for ~ this ~ \c_backslash_str item ~ \msg_line_context:.
5402    }
5403  \msg_new:nnn { enumext } { anskey-empty-arg }
5404    {
5405      Can't ~ store ~ empty ~ content ~ \msg_line_context:.
5406    }
5407  \msg_new:nnn { enumext } { anskey-wrong-place }
5408    {
5409      Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
5410      '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5411    }
5412  \msg_new:nnn { enumext } { anskey-nested }
5413    {
5414      The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
5415    }
5416  \msg_new:nnn { enumext } { anskey-math-mode }
5417    {
5418      #1 ~ can't ~ work ~ in ~ math ~ mode ~ \msg_line_context:.
5419    }
5420  \msg_new:nnn { enumext } { anskey-env-wrong }
5421    {
5422      The ~ environment ~ anskey* ~ cannot ~ use ~ in ~ '#1' ~ \msg_line_context:.
5423    }
5424  \msg_new:nnn { enumext } { anspic-wrong-place }
5425    {
5426      Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
5427      '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5428    }
5429  \msg_new:nnn { enumext } { command-wrong-place }
5430    {
5431      Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
5432      '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
5433    }
5434  \msg_new:nnnn { enumext } { anskey-env-key-unknown }
5435    {
5436      The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5437      'anskey*' ~ and ~ is ~ being ~ ignored.
5438    }
5439    {
5440      The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5441      Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5442    }
5443  \msg_new:nnnn { enumext } { anskey-env-key-value-unknown }
5444    {
5445      The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5446      'anskey*' ~ and ~ is ~ being ~ ignored.
5447    }
5448    {
5449      The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5450      Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5451    }
5452  \msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
5453    { The ~ key ~'#1'~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.}
5454    {
5455      The ~ command ~'\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5456      Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5457    }
```

```
5458  \msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
5459    { The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignor
5460    {
5461      The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5462      Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5463    }
```

Messages used by keyans, keyans* and keyanspic environment.

```
5464  \msg_new:nnn { enumext } { keyans-nested }
5465    {
5466      The ~ environment ~ 'keyans' ~ can't ~ be  ~ nested  ~ \msg_line_context:.
5467    }
5468  \msg_new:nnn { enumext } { keyans-wrong-level }
5469    {
5470      Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:.~ \\
5471      The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5472    }
5473  \msg_new:nnn { enumext } { wrong-place }
5474    {
5475      Wrong ~ place ~ for ~ '#1' ~ environment ~\msg_line_context:.~ \\
5476      '#1' ~ is ~ only ~ found ~ with ~ '#2' ~  in  ~  'enumext.
5477    }
5478  \msg_new:nnn { enumext } { keyanspic-nested }
5479    {
5480      The ~ environment ~ 'keyanspic' ~ can't ~ be  ~ nested~ \msg_line_context:.~.
5481    }
5482  \msg_new:nnn { enumext } { keyanspic-wrong-level }
5483    {
5484      Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:.~ \\
5485      The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5486    }
5487  \msg_new:nnn { enumext } { keyanspic-item-cmd }
5488    {
5489      Can't ~ use  ~ \c_backslash_str item ~ in ~ keyanspic ~ \msg_line_context:.
5490    }
5491  \msg_new:nnnn { enumext } { keyans-unknown-key }
5492    {
5493      The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5494      '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5495    }
5496    {
5497      The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5498      ~ have ~ a ~ key ~ called ~'#1'.\\
5499      Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5500    }
5501  \msg_new:nnnn { enumext } { keyans-unknown-key-value }
5502    {
5503      The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5504      '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5505    }
5506    {
5507      The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5508      ~ have ~ a ~ key ~ called ~'#1'.\\
5509      Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5510    }
```

Message used by unknown ⟨keys⟩ in enumext*. environment.

```
5511  \msg_new:nnnn { enumext } { starred-unknown-key }
5512    {
5513      The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5514      '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5515    }
5516    {
5517      The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5518      ~ have ~ a ~ key ~ called ~'#1'.\\
5519      Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5520    }
5521  \msg_new:nnnn { enumext } { starred-unknown-key-value }
5522    {
5523      The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5524      '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5525    }
```

```
5526       {
5527         The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5528         ~ have ~ a ~ key ~ called ~'#1'.\\
5529         Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5530       }
```

Message used by unknown ⟨*keys*⟩ in enumext environment.

```
5531   \msg_new:nnnn { enumext } { standar-unknown-key }
5532       {
5533         The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_tl
5534         ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5535       }
5536       {
5537         The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5538         ~ have ~ a ~ key ~ called ~'#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5539         Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5540       }
5541   \msg_new:nnnn { enumext } { standar-unknown-key-value }
5542       {
5543         The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_tl
5544         ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5545       }
5546       {
5547         The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5548         ~ have ~ a ~ key ~ called ~'#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5549         Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5550       }
```

Message used by unknown ⟨*keys*⟩ in \foreachkeyans.

```
5551   \msg_new:nnnn { enumext } { for-key-unknown }
5552       { The~key~'#1'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored.}
5553       {
5554         The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5555         Check~that~you~have~spelled~the~key~name~correctly.
5556       }
5557   \msg_new:nnnn { enumext } { for-key-value-unknown }
5558       { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored. }
5559       {
5560         The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5561         Check~that~you~have~spelled~the~key~name~correctly.
5562       }
```

Messages used by \getkeyans command.

```
5563   \msg_new:nnn { enumext } { undefined-storage-anskey }
5564       {
5565         Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
5566       }
```

Messages used by \miniright command.

```
5567   \msg_new:nnn { enumext } { missing-miniright }
5568       {
5569         Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\\
5570         The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
5571       }
5572   \msg_new:nnn { enumext } { wrong-miniright-place }
5573       {
5574         Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
5575         Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
5576       }
5577   \msg_new:nnn { enumext } { wrong-miniright-use }
5578       {
5579         Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
5580         '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
5581       }
5582   \msg_new:nnn { enumext } { wrong-miniright-starred }
5583       {
5584         Can't ~ use ~ \c_backslash_str miniright ~ in ~ starred ~ environments ~ \msg_line_context:.
5585       }
5586   \msg_new:nnn { enumext } { many-miniright-used }
5587       {
5588         Can't ~ use ~ \c_backslash_str miniright ~ more ~ than ~ once ~ \msg_line_context:.
5589       }
```

Messages used by `\setenumextmeta` command.

```
5590 \msg_new:nnn { enumext } { unknown-set }
5591   {
5592     Argument ~ [#1] ~ is ~ unknown ~ by ~ \c_backslash_str setenumextmeta ~ \msg_line_context:.
5593   }
5594 \msg_new:nnn { enumext } { already-defined }
5595   {
5596     The ~ key ~ '#1' ~ is ~ already ~ defined ~ \msg_line_context:.
5597   }
5598 \msg_new:nnn { enumext } { prohibited-unknown }
5599   {
5600     The ~ name ~ 'unknown' ~ can't ~ be ~ chosen~ for ~ a ~ meta ~ key ~ \msg_line_context:.
5601   }
```

Messages used by `enumext*` and `keyans*` environments.

```
5602 \msg_new:nnn { enumext } { nested }
5603   {
5604     The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ \msg_line_con
5605   }
5606 \msg_new:nnn { enumext } { nested-horizontal }
5607   {
5608     The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ in ~ '#1' ~ 
5609   }
5610 \msg_new:nnn { enumext } { item-joined }
5611   {
5612     Items ~ joined ~ (#1) ~ > ~ #2  ~ columns ~\msg_line_context:.
5613   }
5614 \msg_new:nnn { enumext } { item-joined-columns }
5615   {
5616     Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.
5617   }
```

12.51   Finish package

Finish package implementation.

```
5618 \file_input_stop:
5619 ⟨/package⟩
```

13 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.