enumext

ENUMERATE EXERCISE SHEETS

V1.0   2024-09-07[*]

©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:

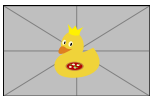

A)   B)
D)   E)

5. Question with image on left side:
   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⟩]
  \item ⟨item content⟩
  \item [⟨custom⟩] ⟨item content⟩
  \item*[⟨symbol⟩][⟨offset⟩] ⟨item content⟩
\end{enumext}
```

```
\begin{enumext*}[⟨keyval list⟩]
  \item ⟨item content⟩
  \item [⟨custom⟩] ⟨item content⟩
  \item*[⟨symbol⟩][⟨offset⟩] ⟨item content⟩
\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{`⟨*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-one = val, key-two = val, ...*⟩`}`
`\setenumextmeta*{`⟨*key name*⟩`}{`⟨*key-one = val, key-two = val, ...*⟩`}`
`\setenumextmeta [`⟨*enumext**⟩`]{`⟨*key name*⟩`}{`⟨*key-one = val, key-two = val, ...*⟩`}`
`\setenumextmeta [`⟨*enumext, level*⟩`]{`⟨*key name*⟩`}{`⟨*key-one = val, key-two = val, ...*⟩`}`

The command `\setenumextmeta` adds a new *"meta-key"* for the environments `enumext` and `enumext*`, the {⟨*key name*⟩} must be different from those defined by the package. 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 ⟨*key = val*⟩ 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 ⟨*key*⟩ is different from ⟨*key=*⟩.

All ⟨*keys*⟩ described in this section are available for the enumext, enumext*, keyans and keyans* environments with the exception of the keys series, resume, resume* 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 ⟨*keys*⟩ related to vertical or horizontal spacing accept a *"skip"* or *"dim"* expression if passed between braces, i.e. you do not need to use \dimeval or \dimexpr to perform calculations.

It should be kept in mind that using any ⟨*key*⟩ that sets 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* ⟩}   default: *by levels*

Sets the ⟨*label*⟩ 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 ⟨*label*⟩ 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 ⟨*label*⟩ as an argument, for example \emph{⟨\alph*⟩} will return an error. For full customization of how ⟨*label*⟩ is displayed use the font or wrap-label keys.

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

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

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

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

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

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

labelwidth = {⟨*rigid length*⟩}   default: *by label*

Sets the *width* of the box containing the current ⟨*label*⟩ 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*⟩}   default: *empty*

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

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

Sets the *font style* for the current ⟨*label*⟩ defined by label key. For example font={\bfseries\small}.

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

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

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

Wraps the *current* ⟨*label*⟩ 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⟩}]        \begin{enumext}[save-ans={⟨store name⟩}]
   \item Text \anskey{answer}                      \item Text \anskey{answer}
   \item Text                                       \item Text
     \begin{keyans}                                   \begin{keyanspic}
       ...                                               ...
     \end{keyans}                                     \end{keyanspic}
\end{enumext}                                     \end{enumext}
```

By executing the key `save-ans={`⟨*store name*⟩`}` the entire structure of the environment (excluding the first level) including the optional 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

A)

B

B)

A

C)

A

D)
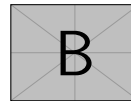
* E)[note]

6.6 Printing stored content

6.6.1 The command \getkeyans

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

The command \getkeyans prints the *"stored content"* in *prop list* {⟨store name⟩} defined by save-ans key in the ⟨position⟩ returned by the show-pos key. The *"stored content"* can only be accessed *after* it is stored, if {⟨store name⟩} does not exist the command will return an error.

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

6.6.2 The command \foreachkeyans

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

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

Options for command

sep = {⟨code⟩}   default: *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.

---

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

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. $2\alpha + 2\delta = 90°$
   II. $\alpha = \delta$
   III. $\angle EDF = 45°$

Ⓐ I only
Ⓑ II only
Ⓒ I and II only

Ⓓ I and III only
Ⓔ I, II, and III

3. Third type of questions

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

Ⓐ value
Ⓑ value
Ⓒ value

Ⓓ value
Ⓔ value

4. Question with image and label below:
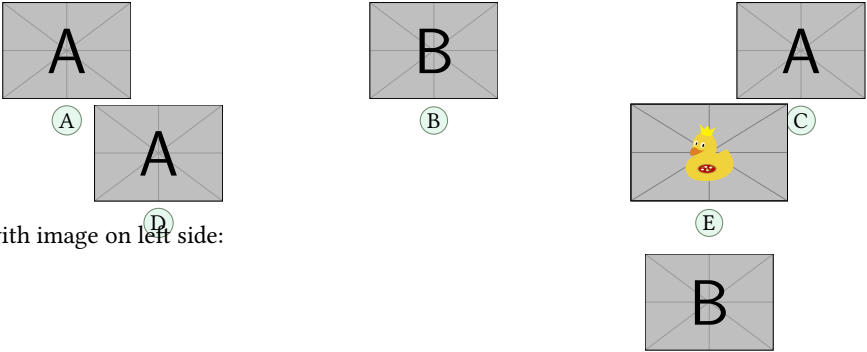
5. Question with image on left side:

Ⓐ value
Ⓑ value
Ⓒ value
Ⓓ correct
Ⓔ 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=2.5em,font=\bfseries`.

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

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

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

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

Description indented by label

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

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

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

SomeThing   A short one-line description.
          This is an entry *without* a label.

Something	A short one-line description.
Something long	A much longer description. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.

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

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

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

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

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

Description with multi-line labels

The `label` key does not accept *multiline material*, this is where the `wrap-label*` 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 2ε: 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 2ε 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 2024-09-07   – 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    {2024-09-07}
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.

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.*)

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.*)

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.*)

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.*)

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.*)

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.*)

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

The variable `\l__enumext_store_active_bool` setting by save-ans key (§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*` (§12.41.1) 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.*)

`\l__enumext_setkey_tmpa_tl`
`\l__enumext_setkey_tmpb_tl`
`\l__enumext_setkey_tmpa_int`
`\l__enumext_setkey_tmpa_seq`
`\l__enumext_setkey_tmpb_seq`

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

`\l__enumext_meta_path_tl`
`\l__enumext_foreach_print_seq`
`\l__enumext_foreach_name_prop_tl`
`\g__enumext_foreach_default_keys_tl`

Internal variables used by the `\printkeyans` command (§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.*)

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

Internal variables used by command `\printkeyans` (§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.*)

`\l__enumext_keyans_pic_body_seq`
`\l__enumext_keyans_pic_width_dim`
`\l__enumext_keyans_pic_above_int`
`\l__enumext_keyans_pic_below_int`
`\l__enumext_keyans_pic_above_skip`

Internal variables used by keyanspic environment (§12.41.2).

```
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
```

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

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.

```
\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
```

```
144  \bool_new:N \l__enumext_check_answers_bool
145  \bool_new:N \g__enumext_check_ans_key_bool
146  \tl_new:N   \l__enumext_check_start_line_env_tl
147  \int_new:N  \g__enumext_check_starred_cmd_int
148  \int_new:N  \g__enumext_item_anskey_int
149  \int_new:N  \g__enumext_item_number_int
150  \bool_new:N \l__enumext_item_number_bool
151  \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.

```
152  \bool_new:N \l__enumext_hyperref_bool
153  \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.

```
154  \tl_new:N \l__enumext_newlabel_arg_one_tl
155  \tl_new:N \l__enumext_newlabel_arg_two_tl
156  \tl_new:N \l__enumext_write_aux_file_tl
157  \cs_set_protected:Npn \__enumext_tmp:n #1
158    {
159      \tl_new:c { l__enumext_label_copy_#1_tl }
160    }
161  \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).

```
162  \int_new:N \g__enumext_footnote_int
163  \seq_new:N \g__enumext_footnote_arg_seq
164  \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.

```
165  \cs_set_protected:Npn \__enumext_tmp:n #1
166    {
167      \bool_new:c { l__enumext_item_starred_#1_bool    }
168      \int_new:c  { l__enumext_item_column_pos_#1_int  }
169      \int_new:c  { g__enumext_item_count_all_#1_int   }
170      \int_new:c  { l__enumext_joined_item_#1_int      }
171      \int_new:c  { l__enumext_joined_item_aux_#1_int  }
172      \int_new:c  { l__enumext_tmpa_#1_int             }
173      \dim_new:c  { l__enumext_tmpa_#1_dim             }
174      \box_new:c  { l__enumext_item_text_#1_box        }
175      \dim_new:c  { l__enumext_joined_width_#1_dim     }
176      \dim_new:c  { l__enumext_item_width_#1_dim       }
177      \tl_new:c   { g__enumext_item_symbol_aux_#1_tl   }
178      \str_new:c  { l__enumext_align_label_#1_str      }
179      \bool_new:c { g__enumext_minipage_active_#1_bool }
180      \box_new:c  { l__enumext_miniright_code_#1_box   }
181      \bool_new:c { g__enumext_minipage_center_#1_bool }
182      \dim_new:c  { g__enumext_minipage_right_#1_dim   }
183      \skip_new:c { g__enumext_minipage_right_#1_skip  }
184    }
185  \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`.

```
186 \clist_const:Nn \c__enumext_all_envs_clist
187   {
188     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
189     {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
190   }
```

(*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).

```
191 \cs_generate_variant:Nn \keys_precompile:nnN { neN }
192 \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.

```
193 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
194   {
195     \hook_gput_code:nnn {begindocument} {enumext} { #1 }
196   }
```

(*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.

```
197 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
198   {
199     \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
200   }
201 \cs_new_protected:Npn \__enumext_before_env:nn #1 #2
202   {
203     \hook_gput_code:nnn {env/#1/before} {enumext} {#2}
204   }
```

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

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

```
205 \cs_new:Nn \__enumext_level:
206   {
207     \int_to_roman:n { \l__enumext_level_int }
208   }
```

(*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?.

```
209 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
210   {
211     \regex_match:nnTF { ^[\+\-]?[\d]+$ } {#1} % $
212       { \prg_return_true: }
213       { \prg_return_false: }
214   }
```

(*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.

```
215 \cs_new_protected:Nn \__enumext_regex_counter_style:
216   {
217     \tl_map_inline:Nn \c__enumext_counter_style_tl
218       {
219         \regex_replace_once:nnN { \c{##1}\* }
220           { \c{##1}\cB{\u{l__enumext_ref_the_count_tl}\cE} } \l__enumext_ref_key_arg_tl
221       }
222   }
```

(*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.

```
223  \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
224    {
225      * ~ #2
226      \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
227        = ~ \use:c { #1_use:c } { l__enumext_#2_#3_#1 } \\
228    }
```

(*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.

```
229  \cs_new_protected:Npn \__enumext_unskip_unkern:
230    {
231      \int_case:nnT { \lastnodetype }
232        {
233          { 11 }
234            {
235                \typeout{SKIIIIIIIIIIIIIIIP}
236                \typeout{\the\lastskip}
237                \unskip
238            }
239          { 12 }
240            {
241                \typeout{KERRRRRRRRRRRRRRRRN}
242                \typeout{\the\lastkern}
243                \unkern
244            }
245        }
246    }
```

(*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.

```
247  \cs_new_protected:Nn \__enumext_is_not_nested:
248    {
249      \str_case:en { \@currenvir }
250        {
251          {enumext}
252            {
253              \tl_set:Nn \l__enumext_envir_name_tl { enumext }
254              \bool_lazy_and:nnT
255                { \bool_not_p:n { \g__enumext_standar_bool } }
256                { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
257                {
258                  \bool_gset_true:N \g__enumext_standar_bool
259                }
260            }
261          {enumext*}
262            {
263              \tl_set:Nn \l__enumext_envir_name_tl { enumext* }
264              \bool_lazy_and:nnT
265                { \bool_not_p:n { \g__enumext_starred_bool } }
266                { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
267                {
268                  \bool_gset_true:N \g__enumext_starred_bool
269                }
270            }
271        }
272    }
```

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.

```
273  \cs_new_protected:Nn \__enumext_is_on_first_level:
274    {
275      \bool_lazy_all:nT
276        {
277          { \bool_if_p:N \g__enumext_standar_bool }
278          { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
279          { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
280        }
281        {
282          \bool_set_true:N \l__enumext_standar_first_bool
283          \bool_set_true:N \l__enumext_anskey_env_bool
284          \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
285          \tl_gset:Ne \g__enumext_start_line_tl
286            {
287              on ~ line ~ \exp_not:V \inputlineno
288            }
289        }
290      \bool_lazy_all:nT
291        {
292          { \bool_if_p:N \g__enumext_starred_bool }
293          { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
294          { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
295        }
296        {
297          \bool_set_true:N \l__enumext_starred_first_bool
298          \bool_set_true:N \l__enumext_anskey_env_bool
299          \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
300          \tl_gset:Ne \g__enumext_start_line_tl
301            {
302              on ~ line ~ \exp_not:V \inputlineno
303            }
304        }
305    }
```

(*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.

```
306  \cs_new_protected:Nn \__enumext_keyans_name_and_start:
307    {
308      \str_case:en { \@currenvir }
309        {
310          {keyans}
311            {
312              \tl_set:Nn \l__enumext_envir_name_tl { keyans }
313              \tl_set:Ne \l__enumext_check_start_line_env_tl
314                {
315                  in ~ 'keyans' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
316                }
317            }
318          {keyans*}
319            {
320              \tl_set:Nn \l__enumext_envir_name_tl { keyans* }
321              \tl_set:Ne \l__enumext_check_start_line_env_tl
322                {
323                  in ~ 'keyans*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
324                }
325            }
326          {keyanspic}
327            {
328              \tl_set:Nn \l__enumext_envir_name_tl { keyanspic }
329              \tl_set:Ne \l__enumext_check_start_line_env_tl
330                {
331                  in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
332                }
333            }
334        }
335    }
```

(*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.

```
336 \cs_new_protected:Nn \__enumext_reset_global_vars:
337   {
338      \__enumext_reset_global_int:
339      \__enumext_reset_global_bool:
340      \__enumext_reset_global_tl:
341   }
342 \cs_new_protected:Nn \__enumext_reset_global_int:
343   {
344      \int_gzero:N \g__enumext_item_number_int
345      \int_gzero:N \g__enumext_item_anskey_int
346      \int_gzero:N \g__enumext_item_answer_diff_int
347   }
348 \cs_new_protected:Nn \__enumext_reset_global_bool:
349   {
350      \bool_gset_false:N \g__enumext_check_ans_key_bool
351      \bool_gset_false:N \g__enumext_standar_bool
352      \bool_gset_false:N \g__enumext_starred_bool
353   }
354 \cs_new_protected:Nn \__enumext_reset_global_tl:
355   {
356      \tl_gclear:N \g__enumext_store_name_tl
357      \tl_gclear:N \g__enumext_start_line_tl
358      \tl_gclear:N \g__enumext_envir_name_tl
359   }
```

(*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.

```
360 \cs_new_protected:Nn \__enumext_log_global_vars:
361   {
362      \msg_log:nneeee { enumext } { prop-seq-int-hook }
363         { \g__enumext_store_name_tl }
364         { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
365         { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
366         { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
367   }
```

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.

```
368 \cs_new_protected:Nn \__enumext_log_answer_vars:
369   {
370      \msg_log:nneee { enumext } { item-answer-hook }
371         { \int_use:N \g__enumext_item_number_int }
372         { \int_use:N \g__enumext_item_anskey_int }
373         { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int} }
374   }
```

(*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.

```
375 \__enumext_at_begin_document:n
376   {
```

```
377    \cs_new_eq:NN  \__enumext_start_list:nn \list
378    \cs_new_eq:NN  \__enumext_stop_list: \endlist
379    \NewCommandCopy \__enumext_item_std:w \item
380  }
```

(*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.

```
381  \__enumext_at_begin_document:n
382    {
383      \cs_new_eq:NN \__enumext_minipage:w \minipage
384      \cs_new_eq:NN \__enumext_endminipage: \endminipage
385    }
```

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

12.7 The 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)

```
386  \cs_new_protected:Nn \__enumext_internal_mini_page:
387    {
388      \int_compare:nNnT { \l__enumext_level_int } = { 0 }
389        {
390          \DeclareDocumentEnvironment{__enumext_mini_page}{ m }
391            {
392              \__enumext_minipage:w [ t ] { ##1 }
393                \legacy_if_gset_false:n { @minipage }
394                \skip_vertical:N \c_zero_skip
395            }
396            {
397                \skip_vertical:N \c_zero_skip
398              \__enumext_endminipage:
399            }
400        }
401    }
```

(*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.

```
402  \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
403  \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".

```
404  \cs_new_protected:Nn \__enumext_after_hyperref:
405    {
406      \IfPackageLoadedTF { hyperref }
407        {
408          \msg_info:nnn { enumext } { package-load } { hyperref }
409          \bool_set_true:N \l__enumext_hyperref_bool
410          \IfHyperBoolean{hyperfootnotes}
411            {
```

```
412        \typeout{hyperfootnotes=true}
413        \bool_set_true:N \l__enumext_footnotes_key_bool
414      }
415      { \typeout{hyperfootnotes=false} }
416    }
417    {  }
```

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.

```
418    \bool_if:NT \l__enumext_footnotes_key_bool
419      {
420        \IfPackageLoadedTF { footnotehyper }
421          {
422            \msg_info:nnn { enumext } { package-load } { footnotehyper }
423          }
424          {
425            \typeout{No ~ footnotehyper ~ load}
426            \typeout{Load ~ and  ~ use  ~ \string\makesavenoteenv{enumext*}}
427            \bool_set_false:N \l__enumext_footnotes_key_bool
428          }
429      }
```

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.

```
430    \bool_if:NTF \l__enumext_hyperref_bool
431      {
432        \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
433        \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
434      }
435      {
436        \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
437        \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
438      }
439    }
```

(*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.

```
440  \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
441    {
442      \protected@write \@auxout { }
443        {
444          \token_to_str:N \newlabel {#1}
445            {
446              {#2}
447              \bool_if:NT \l__enumext_hyperref_bool
448                { { \thepage } {#2} {#1} }
449                { }
450            }
451        }
452      \__enumext_hypertarget:nn {#1} { }
453      \__enumext_phantomsection:
454    }
```

(*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.

```
455  \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.

```
456  \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
457    {
458      \cs_if_exist:cTF { c@ #2 }
459        { \msg_fatal:nnn { enumext } { counters }{ #2 } }
460        {
461          \tl_set:Nn #1 { #2 }
462          \newcounter { #2 }
463        }
464    }
```

(*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.

```
465  \__enumext_define_counters:Nn \l__enumext_counter_i_tl    { enumXi     }
466  \__enumext_define_counters:Nn \l__enumext_counter_ii_tl   { enumXii    }
467  \__enumext_define_counters:Nn \l__enumext_counter_iii_tl  { enumXiii   }
468  \__enumext_define_counters:Nn \l__enumext_counter_iv_tl   { enumXiv    }
469  \__enumext_define_counters:Nn \l__enumext_counter_v_tl    { enumXv     }
470  \__enumext_define_counters:Nn \l__enumext_counter_vi_tl   { enumXvi    }
471  \__enumext_define_counters:Nn \l__enumext_counter_vii_tl  { enumXvii   }
472  \__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.

```
473  \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
474    {
475      \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
476      \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
477    }
478  \__enumext_register_counter_style:Nn \arabic { 0 }
479  \__enumext_register_counter_style:Nn \Alph   { M }
480  \__enumext_register_counter_style:Nn \alph   { m }
481  \__enumext_register_counter_style:Nn \Roman  { VIII }
482  \__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.

```
483  \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
484    {
485      \hbox_set:Nn \l__enumext_label_width_by_box {#2}
486      \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
487    }
488  \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.

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

```
490    {
491      \tl_clear_new:N #1
492      \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
493      \tl_gset_eq:NN \g__enumext_widest_label_tl #1
494      \tl_map_inline:Nn \g__enumext_counter_styles_tl
495        {
496          \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
497          \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
498            { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
499        }
500      \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
501        { \tl_use:N \g__enumext_widest_label_tl }
502      \tl_set_eq:cN { the #2 } #1
503    }
504  \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.

```
505  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
506    {
507      \keys_define:nn { enumext / #1 }
508        {
509          font        .tl_set:c   = { l__enumext_label_font_style_#2_tl },
510          font        .value_required:n = true,
511          labelsep    .dim_set:c  = { l__enumext_labelsep_#2_dim },
512          labelsep    .initial:n  = {0.3333em},
513          labelsep    .value_required:n = true,
514          labelwidth  .dim_set:c  = { l__enumext_labelwidth_#2_dim },
515          labelwidth  .value_required:n = true,
516          wrap-label  .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
517          wrap-label  .initial:n  = {##1},
518          wrap-label  .value_required:n = true,
519          wrap-label* .code:n = {
520                                  \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
521                                  \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
522                                },
523          wrap-label* .value_required:n = true,
524        }
525    }
526  \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.

```
527  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
528    {
529      \keys_define:nn { enumext / #1 }
530        {
531          align .choice:,
532          align / left    .code:n =
533                          {
534                            \tl_clear:c { l__enumext_label_fill_left_#2_tl  }
535                            \tl_set:cn  { l__enumext_label_fill_right_#2_tl } { \hfill }
536                          },
537          align / right   .code:n =
538                          {
539                            \tl_set:cn  { l__enumext_label_fill_left_#2_tl  } { \hfill }
540                            \tl_clear:c { l__enumext_label_fill_right_#2_tl }
541                          },
542          align / center  .code:n =
543                          {
544                            \tl_set:cn { l__enumext_label_fill_left_#2_tl  } { \hfill }
545                            \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
546                          },
```

```
547        align / unknown .code:n =
548                          \msg_error:nneee { enumext } { unknown-choice }
549                            { align } { left, ~ right, ~  center } { \exp_not:n {##1} },
550          align .initial:n  = left,
551          align .value_required:n  = true,
552        }
553    }
554  \clist_map_inline:nn
555    {
556      {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
557    }
558    { \__enumext_tmp:nn #1 }

559  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
560    {
561      \keys_define:nn { enumext / #1 }
562        {
563          align .choice:,
564          align / left     .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
565          align / right    .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
566          align / center   .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
567          align / unknown .code:n =
568                            \msg_error:nneee { enumext } { unknown-choice }
569                              { align } { left, ~ right, ~  center } { \exp_not:n {##1} },
570          align .initial:n = left,
571          align .value_required:n = true,
572        }
573    }
574  \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.

```
575  \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
576    {
577      \keys_define:nn { enumext / #1 }
578        {
579          label .code:n    = {
580                              \__enumext_label_style:cvn { l__enumext_label_#2_tl }
581                                { l__enumext_counter_#2_tl } {##1}
582                              \dim_set_eq:cN  { l__enumext_labelwidth_#2_dim }
583                                \l__enumext_current_widest_dim
584                            },
585          label .initial:n = #3,
586          label .value_required:n = true,
587          ref   .code:n    = \__enumext_standar_ref:n {##1},
588          ref   .value_required:n = true,
589        }
590    }
591  \__enumext_tmp:nnn { level-1 } {   i } { \arabic*.}
592  \__enumext_tmp:nnn { level-2 } {  ii } { (\alph*) }
593  \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
594  \__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.

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

```
596    {
597      \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
598      \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
599        {
600          \msg_error:nnn { enumext } { key-ref-empty } { enumext }
601        }
602        {
603          \tl_set_eq:Nc
604            \l__enumext_ref_the_count_tl { l__enumext_counter_ \__enumext_level: _tl }
605          \__enumext_regex_counter_style:
606          \tl_set_eq:Nc
607            \l__enumext_ref_the_count_tl { l__enumext_the_counter_ \__enumext_level: _tl }
608          \tl_put_right:ce { l__enumext_renew_the_count_ \__enumext_level: _tl }
609            {
610              \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
611                { \exp_not:V \l__enumext_ref_key_arg_tl }
612            }
613        }
614    }
```

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

```
615  \cs_new_protected:Nn \__enumext_standar_ref:
616    {
617      \tl_if_empty:cF { l__enumext_renew_the_count_ \__enumext_level: _tl }
618        {
619          \tl_use:c { l__enumext_renew_the_count_ \__enumext_level: _tl }
620        }
621    }
```

(*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

label
ref
\l__enumext_label_vii_tl
\l__enumext_label_viii_tl

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

```
622  \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
623    {
624      \keys_define:nn { enumext / #1 }
625        {
626          label .code:n     = {
627                               \__enumext_label_style:cvn { l__enumext_label_#2_tl }
628                               { l__enumext_counter_#2_tl } {##1}
629                               \dim_set_eq:cN  { l__enumext_labelwidth_#2_dim }
630                               \l__enumext_current_widest_dim
631                             },
632          label .initial:n = #3,
633          label .value_required:n = true,
634          ref   .code:n     = \__enumext_starred_ref:n {##1},
635          ref   .value_required:n = true,
636        }
637    }
638  \__enumext_tmp:nnn { enumext* } {  vii } { \arabic*.}
639  \__enumext_tmp:nnn { keyans*  } { viii } { \Alph*) }
```

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

__enumext_starred_ref:n
__enumext_starred_ref:

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

```
640  \cs_new_protected:Npn \__enumext_starred_ref:n #1
641    {
642      \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
643      \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
644        {
645          \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
646            {
647              \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
648            }
649            {
650              \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
651              \__enumext_regex_counter_style:
652              \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
653              \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
```

```
654                    {
655                        \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
656                            { \exp_not:V \l__enumext_ref_key_arg_tl }
657                    }
658                }
659            }
660        \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
661            {
662                \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
663                    {
664                        \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
665                    }
666                    {
667                        \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
668                        \__enumext_regex_counter_style:
669                        \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
670                        \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
671                            {
672                                \exp_not:N \renewcommand  { \exp_not:V \l__enumext_ref_the_count_tl }
673                                    { \exp_not:V \l__enumext_ref_key_arg_tl }
674                            }
675                    }
676            }
677    }
```

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.

```
678  \cs_new_protected:Nn \__enumext_starred_ref:
679    {
680        \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
681            {
682                \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
683                    {
684                        \tl_use:N \l__enumext_renew_the_count_vii_tl
685                    }
686            }
687        \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
688            {
689                \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
690                    {
691                        \tl_use:N \l__enumext_renew_the_count_viii_tl
692                    }
693            }
694    }
```

(*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   Here we set the default ⟨*label*⟩ for keyans and keyanspic environment, along with the default value for
ref   labelwidth and ref key. The keyanspic environment use the same ⟨*label*⟩ as the keyans environment.
`\l__enumext_label_v_tl`
`\l__enumext_label_vi_tl`

```
695  \keys_define:nn { enumext / keyans }
696    {
697        label .code:n     = {
698                                \__enumext_label_style:cvn { l__enumext_label_v_tl }
699                                    { l__enumext_counter_v_tl } {#1}
700                                \dim_set_eq:cN  { l__enumext_labelwidth_v_dim }
701                                    \l__enumext_current_widest_dim
702                                \__enumext_label_style:cvn { l__enumext_label_vi_tl }
703                                    { l__enumext_counter_vi_tl } {#1}
704                                \dim_set_eq:cN  { l__enumext_labelwidth_v_dim }
705                                    \l__enumext_current_widest_dim
706                                },
707        label .initial:n = \Alph*),
708        label .value_required:n = true,
709        ref    .code:n    = \__enumext_keyans_ref:n {#1},
710        ref    .value_required:n = true,
711    }
```

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

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

```
712  \cs_new_protected:Npn \__enumext_keyans_ref:n #1
713    {
714      \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
715      \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
716        {
717          \msg_error:nnn { enumext } { key-ref-empty } { keyans }
718        }
719        {
720          \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
721          \__enumext_regex_counter_style:
722          \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
723          \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
724            {
725              \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
726                { \exp_not:V \l__enumext_ref_key_arg_tl }
727            }
728        }
729    }
```

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

```
730  \cs_new_protected:Nn \__enumext_keyans_ref:
731    {
732      \tl_if_empty:NF \l__enumext_renew_the_count_v_tl
733        {
734          \tl_use:N \l__enumext_renew_the_count_v_tl
735        }
736    }
```

(*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.

```
737  \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
738    {
739      \__enumext_if_is_int:nTF { #3 }
740        {
741          \int_set:Nn #2 {#3}
742        }
743        {
744          \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
745            { \int_set:Nn #2 { \int_from_alph:n {#3} } }
746          \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
747            { \int_set:Nn #2  { \int_from_roman:n {#3} } }
748        }
749    }
750  \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.

```
751  \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
752    {
753      \__enumext_if_is_int:nTF {#4}
754        {
755          \setcounter{enumX#1} { #4 }
```

```
756              }
757              {
758                \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
759                  { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
760                \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
761                  { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
762              }
763          \__enumext_label_width_by_box:cv
764            { l__enumext_labelwidth_#1_dim } { l__enumext_label_#1_tl }
765      }
766  \cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }
```

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

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

```
767  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
768    {
769      \keys_define:nn { enumext / #1 }
770        {
771          start* .code:n    = {
772                                \__enumext_start_from:ccn
773                                  { l__enumext_label_#2_tl }
774                                  { l__enumext_start_#2_int } {##1}
775                              },
776          start* .value_required:n = true,
777          start  .code:n    = {
778                                \__enumext_start_from:cce
779                                  { l__enumext_label_#2_tl }
780                                  { l__enumext_start_#2_int } { \int_eval:n {##1} }
781                              },
782          start  .initial:n = 1,
783          start  .value_required:n = true,
784          widest .code:n    = {
785                                \__enumext_widest_from:nccn {#2}
786                                  { l__enumext_label_#2_tl }
787                                  { l__enumext_labelwidth_#2_dim } {##1}
788                              },
789          widest .value_required:n = true,
790        }
791    }
792  \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   Define and set `topsep`, `partopsep`, `parsep`, `itemsep`, `noitemsep` and `nosep` keys for `enumext`, `enumext*`,
partopsep   `keyans` and `keyans*` environments.
parsep
noitemsep
nosep

```
793  \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
794    {
795      \keys_define:nn { enumext / #1 }
796        {
797          topsep    .skip_set:c = { l__enumext_topsep_#2_skip },
798          topsep    .initial:n  = {#3},
799          topsep    .value_required:n = true,
800          partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
801          partopsep .initial:n  = {#4},
802          partopsep .value_required:n = true,
803          parsep    .skip_set:c = { l__enumext_parsep_#2_skip },
804          parsep    .initial:n  = {#5},
805          parsep    .value_required:n = true,
806          itemsep   .skip_set:c = { l__enumext_itemsep_#2_skip },
807          itemsep   .initial:n  = {#6},
808          itemsep   .value_required:n = true,
809          noitemsep .meta:n     = { itemsep = 0pt, parsep = 0pt },
810          noitemsep .value_forbidden:n = true,
811          nosep     .meta:n     = {
812                                    itemsep = 0pt, parsep= 0pt,
813                                    topsep = 0pt, partopsep = 0pt,
814                                  },
```

```
815          nosep    .value_forbidden:n = true,
816        }
817    }
```

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

```
818  \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
819    { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
820    { 4.0pt plus 2.0pt minus 1.0pt }
821  \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
822    { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
823    { 2.0pt plus 1.0pt minus 1.0pt }
824  \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
825    { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
826  \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
827    { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
828  \__enumext_tmp:nnnnnn { keyans  } { v }{ 4.0pt plus 2.0pt minus 1.0pt }
829    { 2.0pt plus 1.0pt minus 1.0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
830    { 2.0pt plus 1.0pt minus 1.0pt }
831  \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
832    { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
833    { 4.0pt plus 2.0pt minus 1.0pt }
834  \__enumext_tmp:nnnnnn { keyans* } { viii } { 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 }
```

(*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*`.

```
837  \cs_set_protected:Npn \__enumext_tmp:n #1
838    {
839      \keys_define:nn { enumext / #1 }
840        {
841          base-fix .bool_set:N = \l__enumext_base_line_fix_bool,
842          base-fix .initial:n  = false,
843          base-fix .value_forbidden:n = true,
844        }
845    }
846  \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)

```
847  \cs_new_protected:Nn \__enumext_nested_base_line_fix:
848    {
849      \bool_lazy_and:nnT
850        { \bool_if_p:N \l__enumext_standar_first_bool }
851        { \bool_if_p:N \l__enumext_base_line_fix_bool }
852        {
853          \mode_leave_vertical:
854          \vspace { -\baselineskip }
855          \keys_set:nn { enumext / level-1 }
856            {
857              topsep = 0pt, above = 0pt, above* = 0pt,
858            }
859        }
860      \bool_lazy_and:nnT
861        { \bool_if_p:N \l__enumext_starred_first_bool }
862        { \bool_if_p:N \l__enumext_base_line_fix_bool }
863        {
864          \mode_leave_vertical:
865          \vspace { -\baselineskip }
866          \keys_set:nn { enumext / enumext* }
867            {
868              topsep = 0pt, above = 0pt, above* = 0pt,
```

```
869            }
870        }
871        \bool_set_false:N \l__enumext_base_line_fix_bool
872    }
```

💣 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.

```
873 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
874    {
875        \keys_define:nn { enumext / #1 }
876          {
877            itemindent    .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
878            itemindent    .value_required:n = true,
879            rightmargin    .dim_set:c = { l__enumext_rightmargin_#2_dim },
880            rightmargin    .value_required:n = true,
881            listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
882            listparindent .value_required:n = true,
883            list-offset    .dim_set:c = { l__enumext_listoffset_#2_dim },
884            list-offset    .value_required:n = true,
885            list-indent    .code:n      =
886                                \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
887                                \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
888            list-indent    .value_required:n = true,
889          }
890    }
891 \clist_map_inline:nn
892    {
893        {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
894    }
895    { \__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.

```
896 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
897    {
898        \keys_define:nn { enumext / #1 }
899          {
900            itemindent    .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
901            itemindent    .value_required:n = true,
902            rightmargin    .dim_set:c = { l__enumext_rightmargin_#2_dim },
903            rightmargin    .value_required:n = true,
904            listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
905            listparindent .value_required:n = true,
906            list-offset    .dim_set:c = { l__enumext_listoffset_#2_dim },
907            list-offset    .value_required:n = true,
908            list-indent    .meta:n      = { list-offset = ##1 },
909            list-indent    .value_required:n = true,
910          }
911    }
912 \clist_map_inline:nn
913    {
914        {enumext*}{vii}, {keyans*}{viii}
915    }
916    { \__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.

```
917 \cs_set_protected:Nn \__enumext_fake_item:
918    {
919        \dim_compare:nNnT
920          { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
921          >
```

```
922        { \c_zero_dim }
923        {
924          \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
925            {
926              \exp_not:N \mode_leave_vertical:
927              \exp_not:n { \skip_horizontal:n }
928                { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
929              \ignorespaces
930            }
931        }
932      }
933  \cs_set_protected:Nn \__enumext_keyans_fake_item:
934    {
935      \dim_compare:nNnT
936        { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
937        {
938          \tl_set:Ne \l__enumext_fake_item_indent_v_tl
939            {
940              \exp_not:N \mode_leave_vertical:
941              \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
942            }
943        }
944    }
945  \cs_set_protected:Nn \__enumext_fake_item_vii:
946    {
947      \dim_compare:nNnT
948        { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
949        {
950          \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
951            {
952              \exp_not:N \mode_leave_vertical:
953              \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
954            }
955        }
956    }
957  \cs_set_protected:Nn \__enumext_fake_item_viii:
958    {
959      \dim_compare:nNnT
960        { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
961        {
962          \tl_set:Ne \l__enumext_fake_item_indent_viii_tl
963            {
964              \exp_not:N \mode_leave_vertical:
965              \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
966            }
967        }
968    }
```

(*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.

```
969  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
970    {
971      \keys_define:nn { enumext / #1 }
972        {
973          show-length .bool_set:c = { l__enumext_show_length_#2_bool },
974          show-length .initial:n  = false,
975        }
976    }
977  \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

```
978  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
979    {
980      \keys_define:nn { enumext / #1 }
981        {
982          before  .tl_set:c   = { l__enumext_before_no_starred_key_#2_tl },
983          before  .value_required:n = true,
984          before* .tl_set:c   = { l__enumext_before_starred_key_#2_tl },
985          before* .value_required:n = true,
986          after   .tl_set:c   = { l__enumext_after_stop_list_#2_tl },
987          after   .value_required:n = true,
988          first   .tl_set:c   = { l__enumext_after_list_args_#2_tl },
989          first   .value_required:n = true,
990        }
991    }
992  \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*⟩}.

```
993  \cs_new_protected:Nn \__enumext_before_args_exec:
994    {
995      \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
996    }
```

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*⟩}}

```
997  \cs_new_protected:Nn \__enumext_before_keys_exec:
998    {
999      \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
1000   }
```

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

```
1001 \cs_new_protected:Nn \__enumext_after_stop_list:
1002   {
1003     \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
1004   }
```

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.

```
1005 \cs_new_protected:Nn \__enumext_after_args_exec:
1006   {
1007     \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
1008   }
```

(*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.

```
1009 \cs_new_protected:Nn \__enumext_before_args_exec_v:
1010   {
1011     \tl_use:N \l__enumext_before_starred_key_v_tl
1012   }
1013 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
1014   {
1015     \tl_use:N \l__enumext_before_no_starred_key_v_tl
1016   }
1017 \cs_new_protected:Nn \__enumext_after_stop_list_v:
1018   {
1019     \tl_use:N \l__enumext_after_stop_list_v_tl
1020   }
1021 \cs_new_protected:Nn \__enumext_after_args_exec_v:
1022   {
1023     \tl_use:N \l__enumext_after_list_args_v_tl
1024   }
```

(*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.

```
1025 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
1026   {
1027     \tl_use:N \l__enumext_before_starred_key_vii_tl
1028   }
1029 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
1030   {
1031     \tl_use:N \l__enumext_before_starred_key_viii_tl
1032   }
1033 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
1034   {
1035     \tl_use:N \l__enumext_before_no_starred_key_vii_tl
1036   }
1037 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
1038   {
1039     \tl_use:N \l__enumext_before_no_starred_key_viii_tl
1040   }
1041 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
1042   {
1043     \tl_use:N \l__enumext_after_stop_list_vii_tl
1044   }
1045 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
1046   {
1047     \tl_use:N \l__enumext_after_stop_list_viii_tl
1048   }
1049 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
1050   {
1051     \tl_use:N \l__enumext_after_list_args_vii_tl
1052   }
1053 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
1054   {
1055     \tl_use:N \l__enumext_after_list_args_viii_tl
1056   }
```

(*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.

```
1057 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1058   {
1059     \keys_define:nn { enumext / #1 }
1060       {
1061         mini-env    .dim_set:c  = { l__enumext_minipage_right_#2_dim },
1062         mini-env    .value_required:n = true,
1063         mini-sep    .dim_set:c  = { l__enumext_minipage_hsep_#2_dim },
1064         mini-sep    .initial:n  = 0.3333em,
1065         mini-sep    .value_required:n = true,
1066         columns-sep .dim_set:c  = { l__enumext_columns_sep_#2_dim },
1067         columns-sep .value_required:n = true,
1068         columns     .int_set:c  = { l__enumext_columns_#2_int },
1069         columns     .initial:n  = 1,
1070         columns     .value_required:n = true,
1071       }
1072   }
1073 \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.

```
1074 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1075   {
1076     \keys_define:nn { enumext / #1 }
1077       {
1078         mini-right  .tl_gset:c = { g__enumext_miniright_code_#2_tl },
1079         mini-right  .value_required:n = true,
1080         mini-right* .code:n    = {
```

```
1081                            \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
1082                            \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1083                        },
1084            mini-right* .value_required:n = true,
1085        }
1086    }
1087 \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*.

```
1088 \cs_new_protected:Nn \__enumext_multi_set_vskip:
1089    {
1090      \skip_set:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1091        {
1092          \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1093        }
1094      \skip_set:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1095        {
1096          \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1097        }
1098      \__enumext_add_pre_parsep:
1099    }
```

(*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*.

```
1100 \cs_new_protected:Nn \__enumext_add_pre_parsep:
1101    {
1102      \int_case:nn { \l__enumext_level_int }
1103        {
1104          { 2 }{
1105              \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1106                {
1107                  \skip_add:Nn \l__enumext_multicols_above_ii_skip { \l__enumext_parsep_i_skip }
1108                }
1109            }
1110          { 3 }{
1111              \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1112                {
```

```
1113                    \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_ii_skip
1114                }
1115            }
1116        { 4 }{
1117                \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1118                {
1119                    \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip
1120                }
1121            }
1122        }
1123    }
```

(*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⟩*.

```
1124 \cs_new_protected:Nn \__enumext_multi_addvspace:
1125    {
1126        \__enumext_multi_set_vskip:
1127        \mode_if_vertical:T
1128            {
1129                \skip_add:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1130                    {
1131                        \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1132                    }
1133                \skip_add:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1134                    {
1135                        \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1136                    }
1137            }
1138        %%\__enumext_unskip_unkern:
1139        \par\nopagebreak
1140        \addvspace{ \skip_use:c { l__enumext_multicols_above_ \__enumext_level: _skip } }
1141    }
```

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

12.21.2 Adjustment of vertical spaces for multicols in keyans

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

```
1142 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1143    {
1144        \skip_set:Nn \l__enumext_multicols_above_v_skip
1145            {
1146                \l__enumext_topsep_v_skip
1147            }
1148        \skip_set:Nn \l__enumext_multicols_below_v_skip
1149            {
1150                \l__enumext_topsep_v_skip
1151            }
1152    }
1153 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1154    {
1155        \__enumext_keyans_multi_set_vskip:
1156        \mode_if_vertical:T
1157            {
1158                \skip_add:Nn \l__enumext_multicols_above_v_skip
1159                    {
1160                        \skip_use:N \l__enumext_partopsep_v_skip
1161                    }
1162                \skip_add:Nn \l__enumext_multicols_below_v_skip
1163                    {
1164                        \skip_use:N \l__enumext_partopsep_v_skip
1165                    }
1166            }
1167        \__enumext_unskip_unkern:
1168        \par\nopagebreak
1169        \addvspace{ \l__enumext_multicols_above_v_skip }
1170    }
```

(*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.

```
1171 \cs_new_protected:Nn \__enumext_minipage_set_skip:
1172   {
1173     \skip_set:Nn \l__enumext_minipage_right_skip
1174       {
1175         \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1176       }
1177     \mode_if_vertical:T
1178       {
1179         \skip_add:Nn \l__enumext_minipage_right_skip
1180           {
1181             \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1182           }
1183       }
1184     \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:.

```
1185     \skip_set_eq:cN
1186       { l__enumext_multicols_above_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1187     \skip_set_eq:cN
1188       { l__enumext_multicols_below_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1189     \__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.

```
1190     \int_compare:nNnT
1191       { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1192       {
1193         \skip_zero:N \topskip
1194         \skip_set_eq:Nc \multicolsep { l__enumext_multicols_above_ \__enumext_level: _skip }
1195       }
1196   }
```

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*.

```
1197  \cs_new_protected:Nn \__enumext_minipage_add_space:
1198    {
1199      \__enumext_minipage_set_skip:
1200      \__enumext_unskip_unkern:
1201      \mode_if_vertical:TF
1202        {
1203          \nopagebreak\nointerlineskip
1204        }
1205        {
1206          \par\nopagebreak\nointerlineskip
1207          \skip_zero:c { l__enumext_partopsep_ \__enumext_level: _skip }
1208        }
1209      \int_compare:nNnTF
1210        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1211        {
1212          \addvspace{ 0.445\box_ht:N \strutbox }
1213        }
1214        {
1215          \addvspace{ 0.250\box_ht:N \strutbox }
1216        }
1217    }
```

(*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.

```
1218  \cs_new_protected:Nn \__enumext_pre_itemsep_skip:
1219    {
1220      \int_case:nn { \l__enumext_level_int }
1221        {
1222          { 2 }{
1223              \skip_if_eq:nnTF
1224                { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1225                {
1226                  \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1227                  \skip_set:Nn \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1228                }
1229                {
1230                  \dim_compare:nNnT
1231                    { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1232                    {
1233                      \skip_sub:Nn
1234                        \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1235                      \skip_sub:Nn
1236                        \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1237                      \skip_add:Nn
1238                        \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1239                      \skip_add:Nn
1240                        \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1241                    }
1242                  \dim_compare:nNnT
1243                    { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1244                    {
1245                      \skip_set:Nn \l_tmpa_skip
1246                        {
1247                          \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1248                        }
1249                      \skip_sub:Nn
1250                        \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1251                      \skip_sub:Nn
1252                        \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1253                      \skip_add:Nn
1254                        \l__enumext_minipage_after_skip
1255                        { 0.150\box_ht:N \strutbox + \l_tmpa_skip }
1256                      \skip_add:Nn
1257                        \l__enumext_multicols_below_ii_skip
1258                        { 0.350\box_ht:N \strutbox + \l_tmpa_skip }
1259                    }
```

```
1260                          }
1261                      }
1262              { 3 }{
1263                  \skip_if_eq:nnTF
1264                      { \l__enumext_itemsep_ii_skip } { \c_zero_skip }
1265                      {
1266                          \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1267                          \skip_set:Nn \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1268                      }
1269                      {
1270                          \dim_compare:nNnT
1271                              { \l__enumext_itemsep_ii_skip } < { \l__enumext_minipage_after_skip }
1272                              {
1273                                  \skip_sub:Nn
1274                                      \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1275                                  \skip_sub:Nn
1276                                      \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1277                                  \skip_add:Nn
1278                                      \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1279                                  \skip_add:Nn
1280                                      \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1281                              }
1282                          \dim_compare:nNnT
1283                              { \l__enumext_itemsep_ii_skip } > { \l__enumext_minipage_after_skip }
1284                              {
1285                                  \skip_set:Nn \l_tmpa_skip
1286                                      {
1287                                          \l__enumext_itemsep_ii_skip - \l__enumext_minipage_after_skip
1288                                      }
1289                                  \skip_sub:Nn
1290                                      \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1291                                  \skip_sub:Nn
1292                                      \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1293                                  \skip_add:Nn
1294                                      \l__enumext_minipage_after_skip
1295                                      { 0.150\box_ht:N \strutbox + \l_tmpa_skip }
1296                                  \skip_add:Nn
1297                                      \l__enumext_multicols_below_iii_skip
1298                                      { 0.350\box_ht:N \strutbox + \l_tmpa_skip }
1299                              }
1300                      }
1301                  }
1302              { 4 }{
1303                  \skip_if_eq:nnTF { \l__enumext_itemsep_iii_skip } { \c_zero_skip }
1304                      {
1305                          \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1306                          \skip_set:Nn \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1307                      }
1308                      {
1309                          \dim_compare:nNnT
1310                              { \l__enumext_itemsep_iii_skip } < { \l__enumext_minipage_after_skip }
1311                              {
1312                                  \skip_sub:Nn
1313                                      \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1314                                  \skip_sub:Nn
1315                                      \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1316                                  \skip_add:Nn
1317                                      \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1318                                  \skip_add:Nn
1319                                      \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1320                              }
1321                          \dim_compare:nNnT
1322                              { \l__enumext_itemsep_iii_skip } > { \l__enumext_minipage_after_skip }
1323                              {
1324                                  \skip_set:Nn \l_tmpa_skip
1325                                      {
1326                                          \l__enumext_itemsep_iii_skip - \l__enumext_minipage_after_skip
1327                                      }
1328                                  \skip_sub:Nn
1329                                      \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1330                                  \skip_sub:Nn
```

```
1331                            \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1332                          \skip_add:Nn
1333                            \l__enumext_minipage_after_skip
1334                            { 0.150\box_ht:N \strutbox + \l_tmpa_skip }
1335                          \skip_add:Nn
1336                            \l__enumext_multicols_below_iv_skip
1337                            { 0.350\box_ht:N \strutbox + \l_tmpa_skip }
1338                        }
1339                      }
1340                    }
1341                  }
1342              }
```

(*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.

```
1343  \cs_new_protected:Nn \__enumext_keyans_minipage_set_skip:
1344    {
1345      \skip_zero:N \l__enumext_minipage_after_skip
1346      \skip_zero:N \l__enumext_minipage_left_skip
1347      \skip_zero:N \l__enumext_minipage_right_skip
1348      \skip_set:Nn \l__enumext_minipage_right_skip
1349        {
1350          \l__enumext_topsep_v_skip
1351        }
1352      \mode_if_vertical:T
1353        {
1354          \skip_add:Nn \l__enumext_minipage_right_skip
1355            {
1356              \l__enumext_partopsep_v_skip
1357            }
1358        }
1359      \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip
1360      \skip_set_eq:NN \l__enumext_multicols_above_v_skip \l__enumext_minipage_right_skip
1361      \skip_set_eq:NN \l__enumext_multicols_below_v_skip \l__enumext_minipage_right_skip
1362      \__enumext_keyans_pre_itemsep_skip:
1363      \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
1364        {
1365          \skip_zero:N \topskip
1366          \skip_set_eq:NN \multicolsep \l__enumext_minipage_right_skip
1367        }
1368    }
1369  \cs_new_protected:Nn \__enumext_keyans_minipage_add_space:
1370    {
1371      \__enumext_keyans_minipage_set_skip:
1372      \__enumext_unskip_unkern:
1373      \mode_if_vertical:TF
1374        {
1375          \nopagebreak\nointerlineskip
1376        }
1377        {
1378          \par\nopagebreak\nointerlineskip
1379          \skip_zero:N \l__enumext_partopsep_v_skip
1380        }
1381      \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1382        {
1383          \addvspace{ 0.445\box_ht:N \strutbox }
1384        }
1385        {
1386          \addvspace{ 0.250\box_ht:N \strutbox }
1387        }
1388    }
1389  \cs_new_protected:Nn \__enumext_keyans_pre_itemsep_skip:
1390    {
1391      \skip_if_eq:nnTF
1392        { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1393        {
1394          \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
```

```
1395        \skip_set:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1396      }
1397      {
1398        \dim_compare:nNnT
1399          { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1400          {
1401            \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1402            \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1403            \skip_add:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1404            \skip_add:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1405          }
1406        \dim_compare:nNnT
1407          { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1408          {
1409            \skip_set:Nn \l_tmpa_skip
1410              {
1411                \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1412              }
1413            \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1414            \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1415            \skip_add:Nn \l__enumext_minipage_after_skip
1416              { 0.150\box_ht:N \strutbox + \l_tmpa_skip }
1417            \skip_add:Nn \l__enumext_multicols_below_v_skip
1418              { 0.350\box_ht:N \strutbox + \l_tmpa_skip }
1419          }
1420      }
1421    }
```

(*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*.

```
1422 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1423    {
1424      \skip_zero_new:N \l__enumext_minipage_left_skip
1425      \skip_gzero_new:N \g__enumext_minipage_right_skip
1426      \skip_gzero_new:N \g__enumext_minipage_after_skip
1427      \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1428        {
1429          \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1430          \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1431        }
1432        {
1433          \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1434          \skip_gset:Nn \g__enumext_minipage_right_skip
1435            {
1436              \l__enumext_topsep_vii_skip
1437            }
1438          \skip_gset:Nn \g__enumext_minipage_after_skip
1439            {
1440              0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1441            }
1442        }
1443    }
1444 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1445    {
1446      \skip_zero_new:N \l__enumext_minipage_after_skip
1447      \skip_zero_new:N \l__enumext_minipage_left_skip
1448      \skip_zero_new:N \l__enumext_minipage_right_skip
1449      \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1450        {
1451          \skip_set:Nn \l__enumext_minipage_left_skip
1452            {
1453              0.5\box_dp:N \strutbox
1454            }
1455          \skip_set:Nn \l__enumext_minipage_right_skip
1456            {
1457              \l__enumext_partopsep_viii_skip
```

```
1458              }
1459          \skip_set:Nn \l__enumext_minipage_after_skip
1460              {
1461                 1.6\box_dp:N \strutbox
1462              }
1463          }
1464          {
1465             \skip_set:Nn \l__enumext_minipage_left_skip
1466                {
1467                   0.5875\box_dp:N \strutbox
1468                }
1469             \skip_set:Nn \l__enumext_minipage_right_skip
1470                {
1471                   \l__enumext_topsep_viii_skip
1472                }
1473             \skip_set:Nn \l__enumext_minipage_after_skip
1474                {
1475                   0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1476                }
1477          }
1478      }
```

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

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

```
1479 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1480   {
1481      \__enumext_mini_set_vskip_vii:
1482      \par\nopagebreak
1483      \addvspace { \l__enumext_minipage_left_skip }
1484   }
1485 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1486   {
1487      \__enumext_mini_set_vskip_viii:
1488      \par\nopagebreak
1489      \addvspace { \l__enumext_minipage_left_skip }
1490   }
```

(*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 environ-
ment or somewhere inappropriate then we will call the internal functions to execute it in the enumext and
keyans environments.

```
1491 \NewDocumentCommand \miniright { s }
1492   {
1493      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1494         {
1495            \msg_error:nnn { enumext } { wrong-miniright-place }
1496         }
1497   % outside
1498   \bool_lazy_and:nnT
1499      { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
1500      { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
1501      {
1502         \msg_error:nnn { enumext } { wrong-miniright-place }
1503      }
1504   % starred env
1505   \bool_if:NT \l__enumext_starred_bool
1506      {
```

```
1507            \msg_error:nnn { enumext } { wrong-miniright-starred }
1508         }
1509       \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
1510         {
1511            \__enumext_keyans_mini_right_cmd:n {#1}
1512         }
1513         { \__enumext_mini_right_cmd:n {#1} }
1514     }
```

(*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.

```
1515 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1516   {
1517     \dim_compare:nNnTF
1518       { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1519       {
1520         \__enumext_multicols_stop:
1521         \int_compare:nNnT
1522           { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } = { 1 }
1523           {
1524             \par\addvspace{ \l__enumext_minipage_after_skip }
1525           }
1526         \end__enumext_mini_page
1527         \hfill
1528         \__enumext_mini_page{ \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } }
1529           \par\nointerlineskip
1530           \addvspace { \l__enumext_minipage_right_skip }
1531           \bool_if:nF {#1}
1532             {
1533                 \centering
1534             }
1535           \int_gzero:N \g__enumext_minipage_stat_int
1536       }
1537       { \msg_error:nnn { enumext } { wrong-miniright-use } }
1538     % paranoia
1539     \RenewDocumentCommand \miniright { s }
1540       {
1541         \msg_error:nn { enumext } { many-miniright-used }
1542       }
1543   }
```

(*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.

```
1544 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1545   {
1546     \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1547       {
1548         \__enumext_keyans_multicols_stop:
1549         \int_compare:nNnT { \l__enumext_columns_v_int } = { 1 }
1550           {
1551             \par\addvspace{ \l__enumext_minipage_after_skip }
1552           }
1553         \end__enumext_mini_page
1554         \hfill
1555         \__enumext_mini_page{ \l__enumext_minipage_right_v_dim }
1556           \par\nointerlineskip
1557           \addvspace { \l__enumext_minipage_right_skip }
1558           \bool_if:nF {#1}
1559             {
1560                 \centering
1561             }
1562           \int_gzero:N \g__enumext_minipage_stat_int
```

```
1563        }
1564    { \msg_error:nnn { enumext } { wrong-miniright-use } }
1565    % paranoia
1566    \RenewDocumentCommand \miniright { s }
1567      {
1568        \msg_error:nn { enumext } { many-miniright-used }
1569      }
1570  }
```

(*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.

```
1571 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1572   {
1573     \keys_define:nn { enumext / #1 }
1574       {
1575         above  .skip_set:c = { l__enumext_vspace_above_#2_skip },
1576         above  .value_required:n = true,
1577         above* .code:n     = \bool_set_true:c { l__enumext_vspace_a_star_#2_bool }
1578                             \keys_set:nn { enumext / #1 } { above = {##1} },
1579         above* .value_required:n = true,
1580         below  .skip_set:c = { l__enumext_vspace_below_#2_skip },
1581         below  .value_required:n = true,
1582         below* .code:n     = \bool_set_true:c { l__enumext_vspace_b_star_#2_bool }
1583                             \keys_set:nn { enumext / #1 } { below = {##1} },
1584         below* .value_required:n = true,
1585       }
1586   }
1587 \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.

```
1588 \cs_new_protected:Nn \__enumext_vspace_above:
1589   {
1590     \skip_if_eq:nnF
1591       { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1592       {
1593         \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
1594           {
1595             \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1596           }
1597           {
1598             \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1599           }
1600       }
1601   }
```

(*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.

```
1602 \cs_new_protected:Nn \__enumext_vspace_below:
1603   {
1604     \skip_if_eq:nnF
1605       { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1606       {
1607         \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1608           {
1609             \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1610           }
```

```
1611          {
1612              \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1613          }
1614      }
1615  }
```

(*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.

```
1616  \cs_new_protected:Nn \__enumext_vspace_above_v:
1617    {
1618      \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1619        {
1620          \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1621            {
1622              \vspace*{ \l__enumext_vspace_above_v_skip }
1623            }
1624            { \vspace { \l__enumext_vspace_above_v_skip } }
1625        }
1626    }
```

(*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.

```
1627  \cs_new_protected:Nn \__enumext_vspace_below_v:
1628    {
1629      \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1630        {
1631          \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1632            {
1633              \vspace*{ \l__enumext_vspace_below_v_skip }
1634            }
1635            { \vspace { \l__enumext_vspace_below_v_skip } }
1636        }
1637    }
```

(*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.

```
1638  \cs_new_protected:Nn \__enumext_vspace_above_vii:
1639    {
1640      \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1641        {
1642          \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1643            {
1644              \vspace*{ \l__enumext_vspace_above_vii_skip }
1645            }
1646            { \vspace { \l__enumext_vspace_above_vii_skip } }
1647        }
1648    }
1649  \cs_new_protected:Nn \__enumext_vspace_above_viii:
1650    {
1651      \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1652        {
1653          \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1654            {
1655              \vspace*{ \l__enumext_vspace_above_viii_skip }
1656            }
1657            { \vspace { \l__enumext_vspace_above_viii_skip } }
1658        }
1659    }
```

(*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.

```
1660 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1661   {
1662     \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1663       {
1664         \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1665           {
1666             \vspace*{ \l__enumext_vspace_below_vii_skip }
1667           }
1668           { \vspace { \l__enumext_vspace_below_vii_skip } }
1669       }
1670   }
1671 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1672   {
1673     \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1674       {
1675         \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1676           {
1677             \vspace*{ \l__enumext_vspace_below_viii_skip }
1678           }
1679           { \vspace { \l__enumext_vspace_below_viii_skip } }
1680       }
1681   }
```

(*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*.

```
1682 \cs_set_protected:Npn \__enumext_tmp:n #1
1683   {
1684     \keys_define:nn { enumext / #1 }
1685       {
1686         series  .str_set:N = \l__enumext_series_str,
1687         series  .value_required:n = true,
1688         resume  .code:n = \__enumext_resume_series:n {##1},
1689         resume* .code:n = \__enumext_resume_starred:,
1690         resume* .value_forbidden:n = true,
1691       }
1692   }
1693 \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.

```
1694 \cs_new:Npn \__enumext_filter_series:n #1
1695   {
1696     \use:e
1697       {
1698         \keyval_parse:NNn
1699           \__enumext_filter_series_key:n
1700           \__enumext_filter_series_pair:nn {#1}
1701       }
1702   }
```

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.

```
1703 \cs_new:Npn \__enumext_filter_series_key:n #1
1704   {
1705     \str_case:nnF {#1}
1706       {
1707         { resume } {} { resume* } {} { base-fix } {}
1708       }
1709       { , { \exp_not:n {#1} } }
1710   }
```

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.

```
1711  \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1712    {
1713      \str_case:nnF {#1}
1714        {
1715          { series } {} { resume } {} { start } {}
1716          { start* } {}  { save-ans } {} { save-key } {}
1717        }
1718        { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1719    }
```

(*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).

```
1720  \cs_new_protected:Npn \__enumext_parse_series:n #1
1721    {
1722      \str_if_empty:NTF \l__enumext_series_str
1723        {
1724          \bool_if:NF \l__enumext_resume_active_bool
1725            {
1726              \__enumext_resume_last:n {#1}
1727            }
1728        }
1729        {
1730          \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str _tl }
1731          \tl_gset:ce { g__enumext_series_ \l__enumext_series_str _tl }
1732            { \__enumext_filter_series:n {#1} }
1733          \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str _int }
1734            {
1735              \int_new:c { g__enumext_series_ \l__enumext_series_str _int }
1736            }
1737        }
1738    }
```

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.

```
1739  \cs_new_protected:Npn \__enumext_resume_last:n #1
1740    {
1741      \bool_if:NT \l__enumext_standar_first_bool
1742        {
1743          \tl_gclear:N \g__enumext_standar_series_tl
1744          \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1745        }
1746      \bool_if:NT \l__enumext_starred_first_bool
1747        {
1748          \tl_gclear:N \g__enumext_starred_series_tl
1749          \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1750        }
1751    }
```

(*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).

```
1752  \cs_new_protected:Nn \__enumext_resume_save_counter:
1753    {
1754      \bool_if:NT \g__enumext_standar_bool
1755        {
1756          \tl_if_empty:NF \l__enumext_series_str
1757            {
1758              \int_gset_eq:cN
1759                { g__enumext_series_ \l__enumext_series_str _int } \value{enumXi}
1760            }
1761          \tl_if_empty:NTF \l__enumext_resume_name_tl
1762            {
1763              \str_if_empty:NT \l__enumext_series_str
1764                {
1765                  \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1766                }
1767            }
1768            {
1769              \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1770                {
1771                  \int_gset_eq:cN
1772                    { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXi}
1773                }
1774            }
1775          \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1776            {
1777              \int_gset_eq:cN
1778                { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1779            }
1780        }
1781      \bool_if:NT \g__enumext_starred_bool
1782        {
1783          \tl_if_empty:NF \l__enumext_series_str
1784            {
1785              \int_gset_eq:cN
1786                { g__enumext_series_ \l__enumext_series_str _int } \value{enumXvii}
1787            }
1788          \tl_if_empty:NTF \l__enumext_resume_name_tl
1789            {
1790              \str_if_empty:NT \l__enumext_series_str
1791                {
1792                  \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1793                }
1794            }
1795            {
1796              \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1797                {
1798                  \int_gset_eq:cN
1799                    { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXvii}
1800                }
1801            }
1802          \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1803            {
1804              \int_gset_eq:cN
1805                { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1806            }
1807        }
1808    }
```

(*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.

```
1809 \cs_new_protected:Npn \__enumext_resume_series:n #1
1810   {
1811     \tl_if_empty:nTF {#1}
1812       {
1813         \__enumext_resume_counter:n { }
1814       }
1815       {
1816         \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1817           {
1818             \__enumext_resume_counter:n {#1}
1819             \bool_if:NT \g__enumext_standar_bool
1820               {
1821                 \keys_set:nv { enumext / level-1 }
1822                   { g__enumext_series_ \tl_to_str:n {#1} _tl }
1823               }
1824             \bool_if:NT \g__enumext_starred_bool
1825               {
1826                 \keys_set:nv { enumext / enumext* }
1827                   { g__enumext_series_ \tl_to_str:n {#1} _tl }
1828               }
1829           }
1830           {
1831             \bool_if:NT \g__enumext_standar_bool
1832               {
1833                 \msg_error:nnn { enumext } { unknown-series } {#1}
1834               }
1835             \bool_if:NT \g__enumext_starred_bool
1836               {
1837                 \msg_error:nnn { enumext } { unknown-series } {#1}
1838               }
1839           }
1840       }
1841   }
```

(*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`.

```
1842 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1843   {
1844     \bool_set_true:N \l__enumext_resume_active_bool
1845     \tl_set:Nn \l__enumext_resume_name_tl {#1}
1846     \tl_if_empty:NTF \l__enumext_resume_name_tl
1847       {
1848         \__enumext_resume_counter:
1849       }
1850       {
1851         \__enumext_resume_counter_series:
1852       }
1853     \__enumext_resume_counter_save_ans:
1854   }
```

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.

```
1855 \cs_new_protected:Nn \__enumext_resume_counter:
1856   {
1857     \bool_if:NT \g__enumext_standar_bool
```

```
1858        {
1859            \int_gincr:N \g__enumext_resume_int
1860            \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1861        }
1862      \bool_if:NT \g__enumext_starred_bool
1863        {
1864            \int_gincr:N \g__enumext_resume_vii_int
1865            \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1866        }
1867    }
```

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.

```
1868  \cs_new_protected:Nn \__enumext_resume_counter_series:
1869    {
1870      \bool_if:NT \g__enumext_standar_bool
1871        {
1872          \int_set:Nn \l__enumext_start_i_int
1873            {
1874              \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1875            }
1876        }
1877      \bool_if:NT \g__enumext_starred_bool
1878        {
1879          \int_set:Nn \l__enumext_start_vii_int
1880            {
1881              \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1882            }
1883        }
1884    }
```

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.

```
1885  \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1886    {
1887      \bool_lazy_and:nnT
1888        { \bool_if_p:N \l__enumext_standar_first_bool }
1889        { \bool_if_p:N \l__enumext_store_active_bool }
1890        {
1891          \int_set:Nn \l__enumext_start_i_int
1892            {
1893              \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1894            }
1895        }
1896      \bool_lazy_and:nnT
1897        { \bool_if_p:N \l__enumext_starred_first_bool }
1898        { \bool_if_p:N \l__enumext_store_active_bool }
1899        {
1900          \int_set:Nn \l__enumext_start_vii_int
1901            {
1902              \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1903            }
1904        }
1905    }
```

(*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.

```
1906  \cs_new_protected:Nn \__enumext_resume_starred:
1907    {
1908      \bool_if:NT \g__enumext_standar_bool
1909        {
1910          \tl_if_empty:NF \g__enumext_standar_series_tl
1911            {
1912              \__enumext_resume_counter:n { }
```

```
1913                \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1914            }
1915        }
1916    \bool_if:NT \g__enumext_starred_bool
1917        {
1918            \tl_if_empty:NF \g__enumext_starred_series_tl
1919                {
1920                    \__enumext_resume_counter:n { }
1921                    \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1922                }
1923            }
1924    }
```

(*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*.

```
1925 \cs_set_protected:Npn \__enumext_tmp:n #1
1926    {
1927        \keys_define:nn { enumext / #1 }
1928            {
1929                save-ans .code:n = \__enumext_storing_set:n {##1},
1930                save-ans .value_required:n = true,
1931            }
1932    }
1933 \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:.

```
1934 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
1935    {
1936        \msg_term:nnVV { enumext } { save-ans-log }
1937            \g__enumext_envir_name_tl \l__enumext_store_name_tl
1938    }
1939 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
1940    {
1941        \msg_term:nnVV { enumext } { save-ans-log-hook }
1942            \g__enumext_envir_name_tl \g__enumext_store_name_tl
1943    }
```

(*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.

```
1944 \cs_new_protected:Npn \__enumext_storing_set:n #1
1945    {
1946        \tl_set:Ne \l__enumext_store_name_tl {#1}
1947        \tl_if_empty:NTF \l__enumext_store_name_tl
1948            {
1949                \bool_lazy_or:nnT
1950                    { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1951                    {
1952                        \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
1953                    }
1954            }
1955            {
1956                \bool_lazy_or:nnT
```

```
1957            { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1958            {
1959              \__enumext_start_save_ans_msg:
1960              \__enumext_storing_exec:
1961            }
1962         }
1963      }
```

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*.

```
1964  \cs_new_protected:Nn \__enumext_storing_exec:
1965    {
1966      \bool_set_true:N \l__enumext_store_active_bool
1967      \bool_set_true:N \l__enumext_check_answers_bool
1968      \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1969      \__enumext_anskey_env_make:V \l__enumext_store_name_tl
1970      \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1971        {
1972          \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
1973          \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1974        }
1975      \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1976        {
1977          \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
1978          \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1979        }
1980      \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1981        {
1982          \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
1983          \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1984        }
1985    }
```

(*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

check-ans
no-store

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

```
1986 \cs_set_protected:Npn \__enumext_tmp:n #1
1987   {
1988     \keys_define:nn { enumext / #1 }
1989       {
1990         check-ans .bool_set:N = \l__enumext_check_ans_key_bool,
1991         check-ans .initial:n  = false,
1992         check-ans .value_required:n = true,
1993         no-store  .code:n = {
1994                             \bool_set_false:N \l__enumext_check_answers_bool
1995                             \bool_set_false:N \l__enumext_check_ans_key_bool
1996                           },
1997         no-store  .value_forbidden:n = true,
1998       }
1999   }
2000 \clist_map_inline:nn
2001   {
2002     level-1, level-2, level-3, level-4, enumext*
2003   }
2004   { \__enumext_tmp:n {#1} }
```

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

12.25.5 Set-up check answer mechanism

__enumext_check_ans_active:
__enumext_check_ans_level:

The function __enumext_check_ans_active: will first check the state of the variable \l__enumext_-store_name_tl, that is, the save-ans key is active, if so it will check the state of the variable \l__enumext_-check_answers_bool handled by the key no-store and will execute the function __enumext_check_-ans_level: only if *"true"*, i.e. the key no-store is not active.

```
2005 \cs_new_protected:Nn \__enumext_check_ans_active:
2006   {
2007     \tl_if_empty:NF \l__enumext_store_name_tl
2008       {
2009         \bool_if:NT \l__enumext_check_answers_bool
2010           {
2011             \__enumext_check_ans_level:
2012           }
2013       }
2014   }
```

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"* .

```
2015 \cs_new_protected:Nn \__enumext_check_ans_level:
2016   {
2017     \int_case:nn { \l__enumext_level_int }
2018       {
2019         { 1 }{
2020             \bool_lazy_all:nT
2021               {
2022                 { \bool_if_p:N \g__enumext_starred_bool }
2023                 { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
2024               }
2025               {
2026                 \int_gdecr:N \g__enumext_item_number_int
2027                 \bool_set_false:N \l__enumext_item_number_bool
2028               }
2029           }
2030         { 2 }{
2031             \int_gdecr:N \g__enumext_item_number_int
2032             \bool_set_false:N \l__enumext_item_number_bool
2033           }
2034         { 3 }{
2035             \int_gdecr:N \g__enumext_item_number_int
2036             \bool_set_false:N \l__enumext_item_number_bool
2037           }
2038         { 4 }{
2039             \int_gdecr:N \g__enumext_item_number_int
2040             \bool_set_false:N \l__enumext_item_number_bool
```

```
2041                    }
2042              }
```

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.

```
2043         \int_case:nn { \l__enumext_level_h_int }
2044           {
2045             { 1 }{
2046                   \bool_lazy_all:nT
2047                     {
2048                       { \bool_if_p:N \g__enumext_standar_bool }
2049                       { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
2050                     }
2051                     {
2052                       \int_gdecr:N \g__enumext_item_number_int
2053                       \bool_set_false:N \l__enumext_item_number_bool
2054                     }
2055                 }
2056           }
2057       }
```

(*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.

```
2058 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
2059   {
2060     \bool_lazy_and:nnT
2061       { \bool_if_p:N \l__enumext_check_ans_key_bool }
2062       { \bool_if_p:N \g__enumext_standar_bool }
2063       {
2064         \bool_gset_true:N \g__enumext_check_ans_key_bool
2065       }
2066     \bool_lazy_and:nnT
2067       { \bool_if_p:N \l__enumext_check_ans_key_bool }
2068       { \bool_if_p:N \g__enumext_starred_bool }
2069       {
2070         \bool_gset_true:N \g__enumext_check_ans_key_bool
2071       }
2072   }
```

(*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:.

```
2073 \cs_new_protected:Nn \__enumext_item_answer_diff:
2074   {
2075     \int_gset:Nn \g__enumext_item_answer_diff_int
2076       {
2077         \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
2078       }
2079   }
```

(*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:.

```
2080 \cs_new_protected:Nn \__enumext_check_ans_show:
2081   {
2082     \int_case:nn { \g__enumext_item_answer_diff_int }
2083       {
2084         { -1 }{ \__enumext_check_ans_msg_less:     }
2085         {  0 }{ \__enumext_check_ans_msg_same_ok: }
2086         {  1 }{ \__enumext_check_ans_msg_greater: }
2087       }
2088   }
```

```
2089 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
2090   {
2091     \msg_warning:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2092       { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2093   }
2094 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
2095   {
2096     \msg_term:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2097       { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2098   }
2099 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
2100   {
2101     \msg_warning:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2102       { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2103   }
```

(*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:.

```
2104 \cs_new_protected:Nn \__enumext_check_ans_log:
2105   {
2106     \int_case:nn { \g__enumext_item_answer_diff_int }
2107       {
2108         { -1 }{ \__enumext_check_ans_log_msg_less:    }
2109         {  0 }{ \__enumext_check_ans_log_msg_same_ok: }
2110         {  1 }{ \__enumext_check_ans_log_msg_greater: }
2111       }
2112   }
2113 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
2114   {
2115     \msg_log:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2116       { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2117   }
2118 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
2119   {
2120     \msg_log:nneee { enumext } { items-same-answer }  { \g__enumext_store_name_tl }
2121       { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2122   }
2123 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
2124   {
2125     \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2126       { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2127   }
```

(*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.

```
2128 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
2129   {
2130     \int_compare:nNnT
2131       { \g__enumext_check_starred_cmd_int } = { 0 }
2132       {
2133         \msg_warning:nnnV
2134           { enumext } { missing-starred }{ #1 } \l__enumext_check_start_line_env_tl
2135       }
2136     \int_compare:nNnT
2137       { \g__enumext_check_starred_cmd_int } > { 1 }
2138       {
2139         \msg_warning:nnnV
2140           { enumext } { many-starred }{ #1 } \l__enumext_check_start_line_env_tl
2141       }
2142     \int_gzero:N \g__enumext_check_starred_cmd_int
2143     \tl_clear:N \l__enumext_check_start_line_env_tl
2144   }
```

(*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*.

```
2145 \cs_set_protected:Npn \__enumext_tmp:n #1
2146   {
2147     \keys_define:nn { enumext / #1 }
2148       {
2149         wrap-ans   .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
2150         wrap-ans   .initial:n =
2151             {
2152               \fbox{\parbox[t]{\dimeval{\itemwidth -2\fboxsep -2\fboxrule}}{##1}}
2153             },
2154         wrap-ans   .value_required:n = true,
2155         wrap-opt   .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
2156         wrap-opt   .initial:n = [{##1}],
2157         wrap-opt   .value_required:n = true,
2158         save-sep   .tl_set:N  = \l__enumext_store_keyans_item_opt_sep_tl,
2159         save-sep   .initial:n = {, ~ },
2160         save-sep   .value_required:n = true,
2161         mark-ans   .tl_set:N  = \l__enumext_mark_answer_sym_tl,
2162         mark-ans   .initial:n = \textasteriskcentered,
2163         mark-ans   .value_required:n = true,
2164         mark-pos   .choice:,
2165         mark-pos / left     .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2166         mark-pos / right    .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2167         mark-pos / unknown .code:n =
2168                             \msg_error:nneee { enumext } { unknown-choice }
2169                               { mark-pos } { left, ~ right } { \exp_not:n {##1} },
2170         mark-pos   .initial:n = right,
2171         mark-pos   .value_required:n = true,
2172         show-ans   .bool_set:N = \l__enumext_show_answer_bool,
2173         show-ans   .initial:n  = false,
2174         show-ans   .value_required:n = true,
2175         show-pos   .bool_set:N = \l__enumext_show_position_bool,
2176         show-pos   .initial:n  = false,
2177         show-pos   .value_required:n = true,
2178         mark-ref   .tl_set:N   = \l__enumext_mark_ref_sym_tl,
2179         mark-ref   .initial:n  = \textasteriskcentered,
2180         mark-ref   .value_required:n = true,
2181         save-ref   .bool_set:N = \l__enumext_store_ref_key_bool,
2182         save-ref   .initial:n  = false,
2183         save-ref   .value_required:n = true,
2184       }
2185   }
2186 \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.

```
2187 \cs_set_protected:Npn \__enumext_tmp:n #1
2188   {
2189     \keys_define:nn { enumext / #1 }
2190       {
2191         mark-pos .choice:,
2192         mark-pos / left   .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2193         mark-pos / right  .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2194         mark-pos .initial:n = right,
2195         mark-pos .value_required:n  = true,
2196         show-ans .bool_set:N = \l__enumext_show_answer_bool,
2197         show-ans .initial:n  = false,
2198         show-ans .value_required:n = true,
2199         show-pos .bool_set:N = \l__enumext_show_position_bool,
2200         show-pos .initial:n  = false,
2201         show-pos .value_required:n = true,
2202       }
2203   }
2204 \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*⟩.

```
2205 \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2206   {
2207     \bool_if:cF { l__enumext_store_save_key_ \__enumext_level: _bool }
2208       {
2209         \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2210         \tl_set:ce
2211           { l__enumext_store_save_key_ \__enumext_level: _tl }
2212           { \__enumext_filter_save_key:n {#1} }
2213       }
2214   }
2215 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2216   {
2217     \bool_if:NF \l__enumext_store_save_key_vii_bool
2218       {
2219         \tl_clear:N \l__enumext_store_save_key_vii_tl
2220         \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2221       }
2222   }
```

(*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.

```
2223 \cs_set_protected:Npn \__enumext_tmp:n #1
2224   {
2225     \keys_define:nn { enumext / enumext* }
2226       {
2227         save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2228         save-key .value_required:n = true,
2229       }
2230     \keys_define:nn { enumext / #1 }
2231       {
2232         save-key .code:n = \__enumext_parse_save_key:n {##1},
2233         save-key .value_required:n = true,
2234       }
2235   }
2236 \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*.

```
2237 \cs_new_protected:Npn \__enumext_parse_save_key:n #1
2238   {
2239     \bool_set_true:c { l__enumext_store_save_key_ \__enumext_level: _bool }
2240     \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2241     \tl_set:ce
2242       { l__enumext_store_save_key_ \__enumext_level: _tl }
2243       { \__enumext_filter_save_key:n {#1} }
2244   }
```

```
2245 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2246   {
2247     \bool_set_true:N \l__enumext_store_save_key_vii_bool
2248     \tl_clear:N \l__enumext_store_save_key_vii_tl
2249     \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2250   }
```

(*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.

```
2251 \cs_new:Npn \__enumext_filter_save_key:n #1
2252   {
2253     \use:e
2254       {
2255         \keyval_parse:NNn
2256           \__enumext_filter_save_key_key:n
2257           \__enumext_filter_save_key_pair:nn {#1}
2258       }
2259   }
```

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.

```
2260 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2261   {
2262     \str_case:nnF {#1}
2263       {
2264         { resume } {} { resume* } {} { no-store } {} { base-fix } {}
2265       }
2266     { , { \exp_not:n {#1} } }
2267   }
```

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.

```
2268 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2269   {
2270     \str_case:nnF {#1}
2271       {
2272         { series   } {} { resume    } {} { save-ans } {} { save-ref   } {}
2273         { save-key } {} { check-ans } {} { show-ans } {} { show-pos   } {}
2274         { wrap-ans } {} { mark-ans  } {} { wrap-opt } {} { save-sep   } {}
2275         { mark-ref } {} { mini-env  } {} { mini-sep } {} { mini-right } {}
2276         { mini-right* } {}
2277       }
2278     { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2279   }
```

(*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.

```
2280 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2281   {
2282     \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2283       {
2284         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2285       }
2286     { #1 }
2287   }
2288 \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.

```
2289 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2290   {
2291     \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
2292   }
2293 \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.

```
2294 \cs_new_protected:Nn \__enumext_store_level_open:
2295   {
2296     \bool_if:NT \l__enumext_check_answers_bool
2297       {
2298         \tl_if_empty:cTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2299           {
2300             \__enumext_store_addto_seq:n
2301               {
2302                 \item \begin{enumext}
2303               }
2304           }
2305           {
2306             \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2307               {
2308                 \item \begin{enumext} [
2309               }
2310             \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2311               {
2312                 ]
2313               }
2314             \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2315           }
2316       }
2317   }
2318 \cs_new_protected:Nn \__enumext_store_level_close:
2319   {
2320     \bool_if:NT \l__enumext_check_answers_bool
2321       {
2322         \__enumext_store_addto_seq:n { \end{enumext} }
2323       }
2324   }
```

(*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.

```
2325 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2326   {
2327     \bool_if:NT \l__enumext_check_answers_bool
2328       {
2329         \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2330           {
2331             \__enumext_store_addto_seq:n
2332               {
2333                 \item \begin{enumext*}
2334               }
2335           }
2336           {
2337             \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2338               {
2339                 \item \begin{enumext*}[
2340               }
2341             \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
```

```
2342                 {
2343                   ]
2344                 }
2345             \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2346           }
2347       }
2348   }
2349 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2350   {
2351     \bool_if:NT \l__enumext_check_answers_bool
2352       {
2353         \__enumext_store_addto_seq:n { \end{enumext*} }
2354       }
2355   }
```

(*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
```

```
2356 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2357   {
2358     \mode_leave_vertical:
2359     \skip_horizontal:n { -\dim_use:N #2 }
2360     \makebox[0pt][ r ]
2361       {
2362         \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2363           {
2364             \tl_use:N \l__enumext_mark_answer_sym_tl
2365           }
2366       }
2367     \skip_horizontal:n { \dim_use:N #2 }
2368   }
2369 \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.

```
2370 \cs_new_protected:Nn \__enumext_store_internal_ref:
2371   {
2372     \cs_set_protected:Npn \__enumext_tmp:n ##1
2373       {
2374         \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2375         \tl_reverse:c { l__enumext_label_copy_##1_tl }
2376         \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2377         \tl_reverse:c { l__enumext_label_copy_##1_tl }
2378       }
2379     \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2380     \cs_set:Npn \__enumext_tmp:n ##1
2381       { . \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.

```
2382       \bool_lazy_all:nT
2383         {
2384           { \bool_if_p:N \g__enumext_starred_bool }
2385           { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2386         }
2387         {
2388           \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2389             { \tl_use:N \l__enumext_label_copy_vii_tl }
2390         }
```

```
2391        \bool_lazy_all:nT
2392          {
2393            { \bool_not_p:n { \g__enumext_standar_bool } }
2394            { \bool_if_p:N \l__enumext_standar_bool }
2395            { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2396          }
2397          {
2398            \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2399              {
2400                \tl_use:N \l__enumext_label_copy_vii_tl
2401                \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2402              }
2403          }
```

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.

```
2404        \bool_lazy_all:nT
2405          {
2406            { \bool_if_p:N \g__enumext_standar_bool }
2407            { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2408            { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2409          }
2410          {
2411            \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2412              {
2413                \tl_use:N \l__enumext_label_copy_i_tl
2414                \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2415              }
2416          }
2417        \cs_set:Npn \__enumext_tmp:n ##1
2418          { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } . }
2419        \bool_lazy_all:nT
2420          {
2421            { \bool_if_p:N \g__enumext_standar_bool }
2422            { \bool_if_p:N \l__enumext_starred_bool }
2423            { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2424          }
2425          {
2426            \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2427              {
2428                \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2429                \tl_use:N \l__enumext_label_copy_vii_tl
2430              }
2431          }
```

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

```
2432        \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2433          {
2434            \l__enumext_store_name_tl \c_colon_str
2435            \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2436          }
```

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.

```
2437        \tl_put_right:Ne \l__enumext_write_aux_file_tl
2438          {
2439            \__enumext_newlabel:nn
2440              { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2441              { \l__enumext_newlabel_arg_two_tl }
2442          }
2443        \l__enumext_write_aux_file_tl
2444      }
```

(*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*⟩.

```
2445  \cs_new_protected:Npn \__enumext_store_anskey_code:n #1
2446    {
```

```
2447        \int_gincr:N \g__enumext_item_anskey_int
2448        \__enumext_store_addto_prop:n {#1}
2449        \bool_if:NT \l__enumext_store_ref_key_bool
2450          {
2451            \__enumext_store_internal_ref:
2452          }
2453        \__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.

```
2454        \tl_clear:N \l__enumext_store_anskey_arg_tl
2455        \bool_lazy_and:nnT
2456          { \bool_if_p:N \l__enumext_store_columns_break_bool }
2457          { \bool_not_p:n { \l__enumext_starred_bool } }
2458          {
2459            \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2460          }
2461        \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.

```
2462        \bool_lazy_and:nnT
2463          { \bool_not_p:n { \l__enumext_starred_bool } }
2464          { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2465          {
2466            \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2467              {
2468                ( \exp_not:V \l__enumext_store_item_join_int )
2469              }
2470          }
```

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*.

```
2471        \bool_if:NTF \l__enumext_store_item_star_bool
2472          {
2473            \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2474            \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2475              {
2476                \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2477                  {
2478                    [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2479                  }
2480              }
2481            \dim_compare:nT
2482              {
2483                \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2484              }
2485              {
2486                \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2487                  {
2488                    [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2489                  }
2490              }
2491            \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2492          }
2493          {
2494            \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2495          }
```

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*⟩.

```
2496        \bool_lazy_and:nnT
2497          { \bool_if_p:N \l__enumext_store_ref_key_bool }
2498          { \bool_if_p:N \l__enumext_hyperref_bool }
2499          {
2500            \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2501              {
2502                \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2503                  { \exp_not:V \l__enumext_mark_ref_sym_tl }
2504              }
```

```
2505              }
2506          \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2507      }
```

(*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.

```
2508  \cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
2509      {
2510          \par
2511          \bool_if:NTF \l__enumext_starred_bool
2512              {
2513                  \__enumext_print_keyans_box:NN \l__enumext_labelwidth_vii_dim  \l__enumext_labelsep_vii_d
2514              }
2515              {
2516                  \__enumext_print_keyans_box:cc
2517                      { l__enumext_labelwidth_ \__enumext_level: _dim }
2518                      { l__enumext_labelsep_ \__enumext_level: _dim }
2519              }
2520          \__enumext_anskey_wrapper:n { #1 }
2521      }
```

(*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.

```
2522  \cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
2523      {
2524          \bool_if:NT \l__enumext_show_answer_bool
2525              {
2526                  \__enumext_anskey_show_wrap_arg:n { #1 }
2527              }
2528          \bool_if:NT \l__enumext_show_position_bool
2529              {
2530                  \tl_set:Ne \l__enumext_mark_answer_sym_tl
2531                      {
2532                          \group_begin:
2533                          \exp_not:N \normalfont
2534                          \exp_not:N \footnotesize [ \int_eval:n
2535                              {
2536                                  \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2537                              }
2538                          ]
2539                          \group_end:
2540                      }
2541                  \__enumext_anskey_show_wrap_arg:n { #1 }
2542              }
2543      }
```

(*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

```
2544  \keys_define:nn { enumext / anskey }
2545      {
2546          break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2547          break-col .default:n  = true,
2548          break-col .value_forbidden:n = true,
2549          item-join .int_set:N  = \l__enumext_store_item_join_int,
2550          item-join .value_required:n = true,
2551          item-star .bool_set:N = \l__enumext_store_item_star_bool,
2552          item-star .default:n  = true,
```

```
2553      item-star .value_forbidden:n = true,
2554      item-sym* .tl_set:N    = \l__enumext_store_item_symbol_tl,
2555      item-sym* .value_required:n = true,
2556      item-pos* .dim_set:N   = \l__enumext_store_item_symbol_sep_dim,
2557      item-pos* .value_required:n = true,
2558      unknown   .code:n      = { \__enumext_anskey_unknown:n {#1} },
2559    }
```

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.

```
2560  \cs_new_protected:Npn \__enumext_anskey_unknown:n #1
2561    {
2562      \exp_args:NV \__enumext_anskey_unknown:nn \l_keys_key_str {#1}
2563    }
2564  \cs_new_protected:Npn \__enumext_anskey_unknown:nn #1 #2
2565    {
2566      \tl_if_blank:nTF {#2}
2567        {
2568          \msg_error:nnn { enumext } { anskey-cmd-key-unknown } {#1}
2569        }
2570        {
2571          \msg_error:nnnn { enumext } { anskey-cmd-key-value-unknown } {#1} {#2}
2572        }
2573    }
```

(*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.

```
2574  \NewDocumentCommand \anskey { o +m }
2575    {
2576      \__enumext_anskey_safe_outer:
2577      \group_begin:
2578        \bool_if:NT \l__enumext_check_answers_bool
2579          {
2580            \tl_if_novalue:nF {#1}
2581              {
2582                \keys_set:nn { enumext / anskey } {#1}
2583              }
2584            \tl_if_blank:nTF {#2}
2585              {
2586                \msg_error:nn { enumext } { anskey-empty-arg }
2587              }
2588              {
2589                \__enumext_anskey_safe_inner:
2590                \__enumext_store_anskey_code:n {#2}
2591              }
2592          }
2593      \group_end:
2594    }
```

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

12.29.1 Internal functions for the command

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

```
2595  \cs_new_protected:Nn \__enumext_anskey_safe_outer:
2596    {
2597      \bool_if:NF \l__enumext_store_active_bool
2598        {
2599          \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
2600        }
2601      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2602        {
2603          \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
```

```
2604          }
2605      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2606        {
2607          \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans* }
2608        }
2609      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2610        {
2611          \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
2612        }
2613    }
```

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.

```
2614 \cs_new_protected:Nn \__enumext_anskey_safe_inner:
2615    {
2616      \int_incr:N \l__enumext_anskey_level_int
2617      \int_compare:nNnT { \l__enumext_anskey_level_int } > { 1 }
2618        {
2619          \msg_error:nn { enumext } { anskey-nested }
2620        }
2621      \bool_if:NF \l__enumext_item_number_bool
2622        {
2623          \msg_error:nn { enumext } { anskey-unnumber-item }
2624        }
2625      \mode_if_math:T
2626        {
2627          \msg_error:nne { enumext } { anskey-math-mode } { \c_backslash_str anskey }
2628        }
2629    }
```

(*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.

```
2630 \cs_new_protected:Nn \__enumext_undefine_anskey_env:
2631    {
2632      \cs_undefine:c { anskey* }
2633      \cs_undefine:c { endanskey* }
2634      \cs_undefine:c { __scontents_anskey*_env_begin: }
2635      \cs_undefine:c { __scontents_anskey*_env_end: }
2636    }
```

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

```
2637 \__enumext_before_env:nn { enumext }
2638    {
2639      \bool_lazy_and:nnT
2640        { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2641        { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2642        {
2643          \cs_if_free:cF { __scontents_anskey*_env_begin: }
2644            {
2645              \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2646            }
2647        }
```

```
2648    }
2649 \__enumext_before_env:nn { enumext* }
2650    {
2651      \bool_lazy_and:nnT
2652        { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2653        { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2654        {
2655          \cs_if_free:cF { __scontents_anskey*_env_begin: }
2656            {
2657              \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2658            }
2659        }
2660    }
```

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.

```
2661 \__enumext_before_env:nn { anskey* }
2662    {
2663      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2664        {
2665          \msg_error:nnn { enumext } { anskey-env-wrong }{ keyans }
2666        }
2667      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2668        {
2669          \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
2670        }
2671      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2672        {
2673          \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
2674        }
2675      \bool_if:NF \l__enumext_item_number_bool
2676        {
2677          \msg_error:nn { enumext } { anskey-unnumber-item }
2678        }
2679      \mode_if_math:T
2680        {
2681          \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
2682        }
2683    }
```

(*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.

```
2684 \cs_new_protected:Npn \__enumext_anskey_env_make:n #1
2685    {
2686      \bool_if:NT \l__enumext_anskey_env_bool
2687        {
2688          \newenvsc{anskey*}[store-env=#1,print-env=false]
2689          \__enumext_anskey_env_exec:
2690        }
2691    }
2692 \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.

```
2693 \cs_new_protected:Nn \__enumext_anskey_env_define_keys:
2694    {
2695      \keys_define:nn { scontents / scontents }
2696        {
2697          break-col .bool_gset:N = \g__enumext_store_columns_break_bool,
2698          break-col .default:n   = true,
2699          break-col .value_forbidden:n = true,
2700          item-join .int_gset:N  = \g__enumext_store_item_join_int,
2701          item-join .value_required:n = true,
```

```
2702          item-star .bool_gset:N = \g__enumext_store_item_star_bool,
2703          item-star .default:n    = true,
2704          item-star .value_forbidden:n = true,
2705          item-sym* .tl_gset:N    = \g__enumext_store_item_symbol_tl,
2706          item-sym* .value_required:n = true,
2707          item-pos* .dim_gset:N   = \g__enumext_store_item_symbol_sep_dim,
2708          item-pos* .value_required:n = true,
2709          print-env .undefine:,
2710          store-env .undefine:,
2711          write-out .undefine:,
2712          unknown   .code:n       = { \__enumext_anskey_env_unknown:n {##1} },
2713        }
2714    }
```

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.

```
2715 \cs_new_protected:Npn \__enumext_anskey_env_unknown:n #1
2716    {
2717      \exp_args:NV \__enumext_anskey_env_unknown:nn \l_keys_key_str {#1}
2718    }
2719 \cs_new_protected:Npn \__enumext_anskey_env_unknown:nn #1#2
2720    {
2721      \tl_if_blank:nTF {#2}
2722        {
2723          \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
2724        }
2725        {
2726          \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
2727        }
2728    }
```

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.

```
2729 \cs_new_protected:Nn \__enumext_anskey_env_reset_keys:
2730    {
2731      \keys_define:nn { scontents / scontents }
2732        {
2733          break-col .undefine:,
2734          item-join .undefine:,
2735          item-star .undefine:,
2736          item-sym* .undefine:,
2737          item-pos* .undefine:,
2738          write-out .code:n    = {
2739                                  \bool_set_false:N \l__scontents_storing_bool
2740                                  \bool_set_true:N  \l__scontents_writing_bool
2741                                  \tl_set:Nn \l__scontents_fname_out_tl {##1}
2742                                },
2743          write-out .value_required:n = true,
2744          print-env .meta:nn   = { scontents } { print-env = ##1 },
2745          print-env .default:n = true,
2746          store-env .meta:nn   = { scontents } { store-env = ##1 },
2747          unknown   .code:n    = { \__scontents_parse_environment_keys:n {##1} },
2748        }
2749    }
```

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*.

```
2750 \cs_new_protected:Npn \__enumext_rescan_anskey_env:n #1
2751    {
2752      \group_begin:
2753        \int_set:Nn \tex_newlinechar:D { `\^^J }
2754        \__scontents_rescan_tokens:x
2755          {
2756            \endgroup % This assumes \catcode`\\=0... Things might go off otherwise.
2757            #1
2758          }
2759    }
```

(*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*⟩.

```
2760  \cs_new_protected:Nn \__enumext_anskey_env_exec:
2761    {
2762      \__enumext_before_env:nn { anskey* }
2763        {
2764          \__enumext_anskey_env_define_keys:
2765        }
```

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.

```
2766      \hook_if_empty:nF {env/anskey*/after}
2767        {
2768          \hook_gremove_code:nn {env/anskey*/after} { * }
2769        }
2770      \__enumext_after_env:nn { anskey* }
2771        {
2772          \__enumext_anskey_env_save_keys:
2773          \tl_clear:N \l__enumext_store_anskey_env_tl
2774          \tl_clear:N \l__enumext_store_anskey_opt_tl
2775          \bool_if:NT \l__enumext_check_answers_bool
2776            {
2777              \tl_gset:Ne \l__enumext_store_anskey_env_tl
2778                {
2779                  \seq_item:ce { g__scontents_name_ \l__enumext_store_name_tl _seq } { -1 }
2780                }
2781              \regex_match:nVTF
2782                { ^\s* \z | ^\s* \u{c__scontents_hidden_space_str} \z }
2783                \l__enumext_store_anskey_env_tl
2784                {
2785                  \msg_error:nn { enumext } { anskey-empty-arg }
2786                }
2787                {
2788                  \__enumext_anskey_env_store:
2789                }
2790            }
2791          \__enumext_anskey_env_clean_vars:
2792          \__enumext_anskey_env_reset_keys:
2793        }
2794    }
```

🐡 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.

```
2795  \cs_new_protected:Nn \__enumext_anskey_env_save_keys:
2796    {
2797      \bool_lazy_and:nnT
2798        { \bool_if_p:N \g__enumext_store_columns_break_bool }
2799        { \bool_not_p:n { \l__enumext_starred_bool } }
2800        {
2801          \tl_put_left:Ne \l__enumext_store_anskey_opt_tl { ,break-col, }
2802        }
```

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

```
2803      \bool_lazy_and:nnT
2804        { \bool_not_p:n { \l__enumext_starred_bool } }
2805        { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }
2806        {
2807          \tl_put_left::Ne \l__enumext_store_anskey_opt_tl
2808            {
2809              ,item-join = \exp_not:V \g__enumext_store_item_join_int,
2810            }
2811        }
```

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

```
2812    \bool_if:NT \g__enumext_store_item_star_bool
2813      {
2814        \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2815          {
2816            ,item-star,
2817          }
2818        \tl_if_empty:NF \g__enumext_store_item_symbol_tl
2819          {
2820            \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2821              {
2822                ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,
2823              }
2824          }
2825        \dim_compare:nT
2826          {
2827            \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
2828          }
2829          {
2830            \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2831              {
2832                ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
2833              }
2834          }
2835      }
2836  }
```

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`.

```
2837  \cs_new_protected:Nn \__enumext_anskey_env_store:
2838    {
2839      \group_begin:
2840        \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
2841          {
2842            \exp_args:Ne
2843              \__enumext_store_anskey_code:n
2844                {
2845                  \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2846                }
2847          }
2848          {
2849            \keys_set_known:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
2850            \exp_args:Ne
2851              \__enumext_store_anskey_code:n
2852                {
2853                  \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2854                }
2855          }
2856      \group_end:
2857    }
```

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

```
2858  \cs_new_protected:Nn \__enumext_anskey_env_clean_vars:
2859    {
2860      \bool_gset_false:N \g__enumext_store_columns_break_bool
2861      \int_gzero:N       \g__enumext_store_item_join_int
2862      \bool_gset_false:N \g__enumext_store_item_star_bool
2863      \tl_gclear:N       \g__enumext_store_item_symbol_tl
2864      \dim_gzero:N       \g__enumext_store_item_symbol_sep_dim
2865    }
```

(*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.

```
2866  \cs_new_protected:Nn \__enumext_execute_after_env:
2867    {
2868      \int_compare:nNnT { \l__enumext_level_int } = { 0 }
2869        {
2870          \tl_if_empty:NF \g__enumext_store_name_tl
2871            {
2872              \__enumext_stop_save_ans_msg:
2873              \__enumext_item_answer_diff:
2874              \__enumext_log_global_vars:
2875              \__enumext_log_answer_vars:
2876              \bool_if:NTF \g__enumext_check_ans_key_bool
2877                {
2878                  \__enumext_check_ans_show:
2879                }
2880                { \__enumext_check_ans_log: }
2881              \__enumext_undefine_anskey_env:
2882            }
2883          \__enumext_reset_global_vars:
2884        }
2885    }
```

(*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.

```
2886  \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2887    {
2888      \tl_clear:N \l__enumext_store_current_label_tl
2889      \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2890        {
2891          \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
2892        }
2893        {
2894          \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
2895        }
2896      \tl_if_novalue:nF { #1 }
2897        {
2898          % Set save-sep
2899          \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2900            {
2901              \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_o
2902            }
2903          \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2904        }
2905      \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
2906    }
```

(*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: The function __enumext_keyans_store_ref: handles the internal *"label and ref"* system used by the
__enumext_keyans_store_ref_aux_i: save-ref key for \item* and \anspic* commands. First we will create copies of the current ⟨*labels*⟩ and
__enumext_keyans_store_ref_aux_ii: remove the dots "." from them, we do not want to get double dots in our references.

```
2907  \cs_new_protected:Nn \__enumext_keyans_store_ref:
2908    {
2909      \bool_if:NT \l__enumext_store_ref_key_bool
2910        {
2911          \cs_set_protected:Npn \__enumext_tmp:n ##1
2912            {
2913              \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2914              \tl_reverse:c { l__enumext_label_copy_##1_tl }
2915              \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2916              \tl_reverse:c { l__enumext_label_copy_##1_tl }
2917            }
2918          \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2919          \__enumext_keyans_store_ref_aux_i:
2920        }
2921    }
```

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.

```
2922  \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2923    {
2924      \bool_if:NT \g__enumext_starred_bool
2925        {
2926          \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2927        }
2928      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2929        {
2930          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2931            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
2932        }
2933      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2934        {
2935          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2936            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2937        }
2938      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2939        {
2940          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2941            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2942        }
2943      \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2944        {
2945          \l__enumext_store_name_tl \c_colon_str
2946          \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2947        }
2948      \__enumext_keyans_store_ref_aux_ii:
2949    }
```

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.

```
2950  \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2951    {
2952      \tl_put_right:Ne \l__enumext_write_aux_file_tl
2953        {
2954          \__enumext_newlabel:nn
2955            { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2956            { \l__enumext_newlabel_arg_two_tl }
2957        }
2958      \l__enumext_write_aux_file_tl
2959    }
```

(*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.

```
2960  \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
```

```
2961      {
2962        \tl_clear:N \l__enumext_store_current_label_tl
2963        \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2964          {
2965            \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
2966          }
2967          {
2968            \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
2969          }
2970        \tl_if_novalue:nF { #1 }
2971          {
2972            \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2973              {
2974                \tl_put_right:Ne \l__enumext_store_current_label_tl
2975                  {
2976                    \l__enumext_store_keyans_item_opt_sep_tl
2977                  }
2978              }
2979            \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2980          }
2981        \__enumext_keyans_addto_seq_link:
2982      }
```

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.

```
2983  \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2984    {
2985      \bool_lazy_and:nnT
2986        { \bool_if_p:N \l__enumext_store_ref_key_bool }
2987        { \bool_if_p:N \l__enumext_hyperref_bool }
2988        {
2989          \tl_put_right:Ne \l__enumext_store_current_label_tl
2990            {
2991              \hfill \exp_not:N \hyperlink
2992                {
2993                  \exp_not:V \l__enumext_newlabel_arg_one_tl
2994                }
2995                { \exp_not:V \l__enumext_mark_ref_sym_tl }
2996            }
2997        }
2998      \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
2999      \bool_if:NT \l__enumext_check_answers_bool
3000        {
3001          \int_gincr:N \g__enumext_item_anskey_int
3002        }
3003    }
```

(*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.

```
3004  \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
3005    {
3006      \tl_if_novalue:nF { #1 }
3007        {
3008          \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
3009        }
3010      \bool_if:NT \l__enumext_show_answer_bool
3011        {
3012          \__enumext_keyans_show_ans:
3013        }
3014      \bool_if:NT \l__enumext_show_position_bool
3015        {
```

```
3016              \__enumext_keyans_show_pos:
3017          }
3018    }
3019  \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
3020    {
3021      \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
3022        {
3023          \bool_lazy_or:nnT
3024            { \bool_if_p:N \l__enumext_show_answer_bool }
3025            { \bool_if_p:N \l__enumext_show_position_bool }
3026            {
3027              \__enumext_keyans_wrapper_opt:n { \l__enumext_store_current_opt_arg_tl } \c_space_tl
3028            }
3029        }
3030    }
3031  \cs_new_protected:Nn \__enumext_keyans_show_ans:
3032    {
3033      \bool_if:NT \l__enumext_starred_bool
3034        {
3035          \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
3036          \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
3037        }
3038      \tl_put_left:Nn \l__enumext_label_v_tl
3039        {
3040          \__enumext_print_keyans_box:NN
3041            \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3042        }
3043    }
3044  \cs_new_protected:Nn \__enumext_keyans_show_pos:
3045    {
3046      \bool_if:NT \l__enumext_starred_bool
3047        {
3048          \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
3049          \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
3050        }
3051      \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
3052        {
3053          \tl_set:Ne \l__enumext_mark_answer_sym_tl
3054            {
3055              \group_begin:
3056              \exp_not:N \normalfont
3057              \exp_not:N \footnotesize [ \int_eval:n
3058                {
3059                  \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3060                }
3061                ]
3062              \group_end:
3063            }
3064        }
3065        {
3066          \tl_set:Ne \l__enumext_mark_answer_sym_tl
3067            {
3068              \group_begin:
3069              \exp_not:N \normalfont
3070              \exp_not:N \footnotesize [ \int_eval:n
3071                {
3072                  \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
3073                }
3074                ]
3075              \group_end:
3076            }
3077        }
3078      \tl_put_left:Nn \l__enumext_label_v_tl
3079        {
3080          \__enumext_print_keyans_box:NN
3081            \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3082        }
3083    }
```

(*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.

```
3084  \cs_new_protected:Npn \__enumext_default_item:n #1
3085    {
3086      \tl_if_novalue:nTF {#1}
3087        {
3088          \bool_if:NT \l__enumext_check_answers_bool
3089            {
3090              \int_gincr:N \g__enumext_item_number_int
3091              \bool_set_true:N \l__enumext_item_number_bool
3092            }
3093          \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
3094          \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3095        }
3096        {
3097          \bool_set_eq:cc
3098            { l__enumext_wrap_label_ \__enumext_level: _bool }
3099            { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
3100          \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl
3101        }
3102    }
```

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

__enumext_starred_item:nn
__enumext_item_star_exec:

The \item*, \item*[⟨symbol⟩] and \item*[⟨symbol⟩][⟨offset⟩] works like the *numbered* \item, but placing a ⟨symbol⟩ to the "*left*" of the ⟨label⟩ separated from it by the value the second optional argument ⟨offset⟩.

#1: \l__enumext_item_symbol_X_tl
#2: \l__enumext_item_symbol_sep_X_dim

First we will make a copy of \l__enumext_item_symbol_X_tl which is set by the key item-sym* or passed as "*first*" optional argument in the global variable \g__enumext_item_symbol_aux_tl, followed by setting the variable \l__enumext_item_symbol_sep_X_dim set by the key item-pos* or by the "*second*" optional argument, then we will see the state of the variable \l__enumext_check_answers_bool set by the key no-store, set the boolean variable \l__enumext_wrap_label_X_bool to "true" for the key wrap-label and execute __enumext_item_std:w and the key itemindent.

```
3103  \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
3104    {
3105      \tl_if_novalue:nTF {#1}
3106        {
3107          \tl_gset_eq:Nc
3108            \g__enumext_item_symbol_aux_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
3109        }
3110        {
3111          \tl_gset:Nn \g__enumext_item_symbol_aux_tl {#1}
3112        }
3113      \tl_if_novalue:nTF {#2}
3114        {
3115          \dim_set_eq:cc
3116            { l__enumext_item_symbol_sep_ \__enumext_level: _dim }
3117            { l__enumext_labelsep_ \__enumext_level: _dim }
3118        }
3119        {
3120          \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
3121        }
3122      \bool_if:NT \l__enumext_check_answers_bool
3123        {
3124          \int_gincr:N \g__enumext_item_number_int
3125          \bool_set_true:N \l__enumext_item_number_bool
```

```
3126              }
3127          \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
3128          \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3129      }
```

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

```
3130  \cs_new_protected:Nn \__enumext_item_star_exec:
3131      {
3132          \tl_if_empty:cF { l__enumext_item_symbol_ \__enumext_level: _tl }
3133              {
3134                  \mode_leave_vertical:
3135                  \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3136                  \hbox_overlap_left:n { \g__enumext_item_symbol_aux_tl }
3137                  \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3138              }
3139      }
```

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

__enumext_redefine_item:
__enumext_make_label

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

```
3140  \cs_new_protected:Nn \__enumext_redefine_item:
3141      {
3142          \RenewDocumentCommand \item { s o o }
3143              {
3144                  \bool_if:nTF {##1}
3145                      {
3146                          \__enumext_starred_item:nn {##2} {##3}
3147                      }
3148                      { \__enumext_default_item:n {##2} }
3149              }
3150      }
```

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

```
3151  \cs_new_protected:Nn \__enumext_make_label:
3152      {
3153          \RenewDocumentCommand \makelabel { m }
3154              {
3155                  \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
3156                  \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3157                  \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3158                      {
3159                          \__enumext_item_star_exec:
3160                          \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3161                      }
3162                      { ##1 }
3163                  \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
3164                  \tl_gclear:N \g__enumext_item_symbol_aux_tl
3165              }
3166      }
```

(*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*
item-pos*

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

```
3167  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
3168      {
3169          \keys_define:nn { enumext / #1 }
3170              {
3171                  item-sym* .tl_set:c  = { l__enumext_item_symbol_#2_tl },
3172                  item-sym* .value_required:n = true,
3173                  item-sym* .initial:n  = {$\star$},
3174                  item-pos* .dim_set:c  = { l__enumext_item_symbol_sep_#2_dim },
3175                  item-pos* .value_required:n = true,
3176              }
```

```
3177    }
3178  \clist_map_inline:nn
3179    {
3180      {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
3181    }
3182    { \__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.

```
3183  \cs_set_protected:Npn \__enumext_tmp:n #1
3184    {
3185      \keys_define:nn { enumext / #1 }
3186        {
3187          unknown .code:n = { \__enumext_keyans_unknown_keys:n {##1} }
3188        }
3189    }
3190  \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }
```

Internal functions for handling unknown key.

```
3191  \cs_new_protected:Npn \__enumext_keyans_unknown_keys:n #1
3192    {
3193      \exp_args:NV \__enumext_keyans_unknown_keys:nn \l_keys_key_str {#1}
3194    }
3195  \cs_new_protected:Npn \__enumext_keyans_unknown_keys:nn #1#2
3196    {
3197      \tl_if_blank:nTF {#2}
3198        {
3199          \msg_error:nnn { enumext } { keyans-unknown-key } {#1}
3200        }
3201        {
3202          \msg_error:nnnn { enumext } { keyans-unknown-key-value } {#1} {#2}
3203        }
3204    }
```

(*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.

```
3205  \keys_define:nn { enumext / enumext* }
3206    {
3207      unknown .code:n = { \__enumext_starred_unknown_keys:n {#1} }
3208    }
```

Internal functions for handling unknown key.

```
3209  \cs_new_protected:Npn \__enumext_starred_unknown_keys:n #1
3210    {
3211      \exp_args:NV \__enumext_starred_unknown_keys:nn \l_keys_key_str {#1}
3212    }
3213  \cs_new_protected:Npn \__enumext_starred_unknown_keys:nn #1#2
3214    {
3215      \tl_if_blank:nTF {#2}
3216        {
3217          \msg_error:nnn { enumext } { starred-unknown-key } {#1}
3218        }
3219        {
3220          \msg_error:nnnn { enumext } { starred-unknown-key-value } {#1} {#2}
3221        }
3222    }
```

(*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.

```
3223 \cs_set_protected:Npn \__enumext_tmp:n #1
3224   {
3225     \keys_define:nn { enumext / #1 }
3226       {
3227         unknown .code:n = { \__enumext_standar_unknown_keys:n {##1} }
3228       }
3229   }
3230 \clist_map_inline:nn { level-1,level-2,level-3,level-4 } { \__enumext_tmp:n {#1} }
```

Internal functions for handling unknown key.

```
3231 \cs_new_protected:Npn \__enumext_standar_unknown_keys:n #1
3232   {
3233     \exp_args:NV \__enumext_standar_unknown_keys:nn \l_keys_key_str {#1}
3234   }
3235 \cs_new_protected:Npn \__enumext_standar_unknown_keys:nn #1#2
3236   {
3237     \tl_if_blank:nTF {#2}
3238       {
3239         \msg_error:nnn { enumext } { standar-unknown-key } {#1}
3240       }
3241       {
3242         \msg_error:nnnn { enumext } { standar-unknown-key-value } {#1} {#2}
3243       }
3244   }
```

(*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.

```
3245 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
3246   {
3247     \tl_if_novalue:nTF { #1 }
3248       {
3249         \bool_set_true:N \l__enumext_wrap_label_v_bool
3250         \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
3251       }
3252       {
3253         \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
3254         \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
3255       }
3256   }
```

(*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*⟩.

```
3257 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
3258   {
3259     \tl_set_eq:NN \l__enumext_store_current_label_tmp_tl \l__enumext_label_v_tl
3260     \__enumext_keyans_show_left:n { #1 }
3261     \bool_set_true:N \l__enumext_wrap_label_v_bool
3262     \__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*⟩.

```
3263     \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_store_current_label_tmp_tl
3264     \__enumext_keyans_addto_prop:n { #1 }
3265     \__enumext_keyans_store_ref:
3266     \__enumext_keyans_addto_seq:n { #1 }
3267     \int_gincr:N \g__enumext_check_starred_cmd_int
3268   }
```

(*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.

```
3269  \cs_new_protected:Nn \__enumext_keyans_redefine_item:
3270    {
3271      \RenewDocumentCommand \item { s o }
3272        {
3273          \bool_if:nTF {##1}
3274            {
3275              \peek_remove_spaces:n
3276                {
3277                  \__enumext_keyans_starred_item:n {##2}
3278                }
3279            }
3280            {
3281              \__enumext_keyans_default_item:n {##2}
3282            }
3283        }
3284    }
```

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

```
3285  \cs_new_protected:Nn \__enumext_keyans_make_label:
3286    {
3287      \RenewDocumentCommand \makelabel { m }
3288        {
3289          \tl_use:N \l__enumext_label_fill_left_v_tl
3290          \tl_use:N \l__enumext_label_font_style_v_tl
3291          \bool_if:NTF \l__enumext_wrap_label_v_bool
3292            {
3293              \__enumext_wrapper_label_v:n { ##1 }
3294            }
3295            { ##1 }
3296          \tl_use:N \l__enumext_label_fill_right_v_tl
3297        }
3298    }
```

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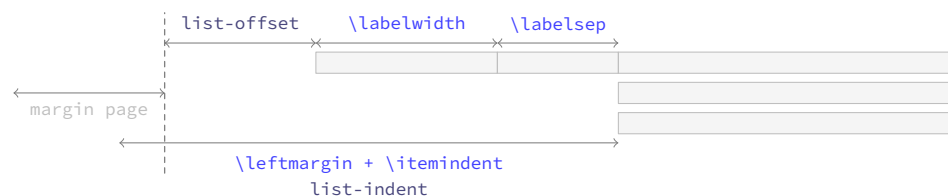
Figure 9: Representation of standard horizontal lengths in list environment.

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.
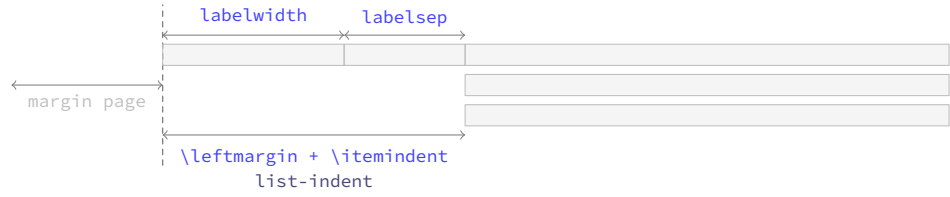
Figure 11: Default horizontal lengths in enumext.

```
#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).

```
3299  \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
3300    {
3301      \dim_compare:nNnT { #1 } < { \c_zero_dim }
3302        {
3303          \msg_warning:nnnV { enumext } { width-non-positive }{ labelwidth }{ #1 }
3304          \dim_set:Nn #1 { \dim_abs:n { #1 } }
3305        }
3306      \dim_compare:nNnT { #2 } < { \c_zero_dim }
3307        {
3308          \msg_warning:nnnV { enumext } { width-negative }{ labelsep }{ #2 }
3309          \dim_set:Nn #2 { \dim_abs:n { #2 } }
3310        }
```

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

```
3311        \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2} }
```

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

```
3312        \dim_compare:nNnTF { #4 } < { \c_zero_dim }
3313          {
3314          \dim_set:Nn #6 { #1 + #2 - #4}
3315          \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3316          }
3317          {
3318          \dim_compare:nNnT { #4 } = { #1 + #2 }
3319            { \dim_set:Nn #6 { \c_zero_dim } }
3320          \dim_compare:nNnT { #4 } < { #1 + #2 }
3321            { \dim_set:Nn #6 { #1 + #2 - #4} }
3322          \dim_compare:nNnT { #4 } > { #1 + #2 }
3323            {
3324            \dim_set:Nn #6 { -#1 - #2 + #4}
3325            \dim_set:Nn #6 { #6*-1}
3326            }
3327          \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3328          }
3329    }
3330  \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.

```
3331 \cs_set_protected:Npn \__enumext_tmp:n #1
3332   {
3333     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3334       {
3335         \__enumext_calc_hspace:ccccccc
3336           { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3337           { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3338           { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3339           { l__enumext_leftmargin_tmp_#1_bool }
3340         \clist_map_inline:nn
3341           { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3342           { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
3343         \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3344           { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
3345         \usecounter { enumX#1 }
3346         \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
3347         \str_if_eq:nnTF {#1} { v }
3348           {
3349             \__enumext_keyans_redefine_item:
3350             \__enumext_keyans_make_label:
3351             \__enumext_keyans_ref:
3352             \__enumext_keyans_fake_item:
3353             \bool_if:cT { l__enumext_show_length_#1_bool }
3354               {
3355                 \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3356               }
3357           }
3358           {
3359             \__enumext_redefine_item:
3360             \__enumext_make_label:
3361             \__enumext_standar_ref:
3362             \__enumext_fake_item:
3363             \bool_if:cT { l__enumext_show_length_#1_bool }
3364               {
3365                 \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \l__enumext_level_i
3366               }
3367           }
3368       }
3369   }
3370 \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.

```
3371 \cs_set_protected:Npn \__enumext_tmp:n #1
3372   {
3373     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3374       {
3375         \bool_set_true:c { l__enumext_leftmargin_tmp_#1_bool }
3376         \dim_zero:c { l__enumext_leftmargin_tmp_#1_dim }
3377         \__enumext_calc_hspace:ccccccc
3378           { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3379           { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3380           { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3381           { l__enumext_leftmargin_tmp_#1_bool }
3382         \clist_map_inline:nn
3383           { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3384           { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
3385         \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3386           { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
3387         \skip_set_eq:Nc \parsep  { l__enumext_itemsep_#1_skip }
3388         \skip_zero:N \partopsep
3389         \usecounter { enumX#1 }
3390         \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
```

```
3391            \__enumext_starred_ref:
3392            \str_if_eq:nnTF {#1} { vii }
3393              {
3394                \__enumext_fake_item_vii:
3395                \bool_if:cT { l__enumext_show_length_vii_bool }
3396                  { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3397              }
3398              {
3399                \__enumext_fake_item_viii:
3400                \bool_if:cT { l__enumext_show_length_#1_bool }
3401                  { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3402              }
3403          }
3404      }
3405  \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.

```
3406  \NewDocumentEnvironment{enumext}{ O{} }
3407    {
3408      \__enumext_safe_exec:
3409      \__enumext_parse_keys:n {#1}
3410      \__enumext_before_list:
3411      \__enumext_start_store_level:
3412      \__enumext_start_list:nn
3413        { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
3414        {
3415          \use:c { __enumext_list_arg_two_ \__enumext_level: : }
3416          \__enumext_before_keys_exec:
3417        }
3418      \__enumext_set_item_width:
3419      \__enumext_after_args_exec:
3420    }
3421    {
3422      \__enumext_stop_list:
3423      \__enumext_stop_store_level:
3424      \__enumext_after_list:
3425    }
```

(*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.

```
3426  \cs_new_protected:Nn \__enumext_set_item_width:
3427    {
3428      \dim_set:Nn \itemwidth
3429        {
3430          \linewidth
3431        }
3432      \dim_compare:nT
3433        {
3434          \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim } != \c_zero_dim
3435        }
3436        {
3437          \dim_sub:Nn \itemwidth
3438            {
3439              \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3440            }
3441        }
3442    }
```

(*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"*.

```
3443  \cs_new_protected:Nn \__enumext_safe_exec:
3444    {
3445      \__enumext_internal_mini_page:
3446      \__enumext_is_not_nested:
3447      \int_incr:N \l__enumext_level_int
3448      \int_compare:nNnT { \l__enumext_level_int } > { 4 }
3449        { \msg_fatal:nn { enumext } { list-too-deep } }
3450      \bool_set_true:N \l__enumext_standar_bool
3451      \bool_set_false:N \l__enumext_starred_bool
3452      \__enumext_is_on_first_level:
3453    }
```

(*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.

```
3454  \cs_new_protected:Npn \__enumext_parse_keys:n #1
3455    {
3456      \tl_if_novalue:nF {#1}
3457        {
3458          \str_clear:N \l__enumext_series_str
3459          \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
3460            {
3461              \keys_set:nn { enumext / level-1 } {#1}
3462              \__enumext_parse_series:n {#1}
3463              \__enumext_nested_base_line_fix:
3464            }
3465            {
3466              \exp_args:Ne \keys_set:nn
3467                { enumext / level-\int_use:N \l__enumext_level_int } {#1}
3468            }
3469          \__enumext_store_active_keys:n {#1}
3470        }
3471    }
```

(*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*`.

```
3472  \cs_new_protected:Nn \__enumext_start_store_level:
3473    {
3474      \bool_lazy_all:nT
3475        {
3476          { \bool_if_p:N \l__enumext_store_active_bool }
3477          { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3478          { \bool_if_p:N \g__enumext_standar_bool }
3479        }
3480        {
3481          \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3482            {
3483              \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3484              \__enumext_store_level_open:
3485            }
3486        }
```

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

```
3487      \bool_lazy_all:nT
3488        {
3489          { \bool_if_p:N \l__enumext_store_active_bool }
3490          { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3491          { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
3492        }
3493        {
3494          \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3495            {
```

```
3496                    \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3497                    \__enumext_store_level_open:
3498                }
3499            }
3500    }
```

Close the stored structure.

```
3501 \cs_new_protected:Nn \__enumext_stop_store_level:
3502    {
3503        \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
3504            {
3505                \__enumext_store_level_close:
3506            }
3507    }
```

(*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.

```
3508 \cs_new_protected:Nn \__enumext_before_list:
3509    {
3510        \__enumext_vspace_above:
3511        \__enumext_before_args_exec:
3512        \__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.

```
3513        \dim_compare:nNnT
3514            { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3515            {
3516                \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
3517                    {
3518                        \linewidth
3519                        - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
3520                        - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
3521                    }
```

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.

```
3522            \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
3523            \int_gincr:N \g__enumext_minipage_stat_int
3524            \__enumext_minipage_add_space:
3525            \__enumext_mini_page{ \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
3526            }
3527        \__enumext_multicols_start:
3528    }
```

(*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.

```
3529 \cs_new_protected:Nn \__enumext_multicols_start:
3530    {
3531        \int_compare:nNnT
3532            { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3533            {
3534                \dim_compare:nNnT
3535                    { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3536                    {
3537                        \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
3538                            {
3539                                ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
```

```
3540                    + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
3541                  ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
3542                  - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3543              }
3544          }
3545          \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
3546          \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3547            {
3548                \dim_zero:N \columnseprule
3549            }
```

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.

```
3550          \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3551            {
3552                \skip_zero:N \multicolsep
3553                \__enumext_multi_addvspace:
3554            }
3555          \raggedcolumns
3556          \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
3557      }
3558    }
```

(*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.

```
3559 \cs_new_protected:Nn \__enumext_multicols_stop:
3560   {
3561     \int_compare:nNnT
3562       { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3563       {
3564         \end{multicols}
3565         \__enumext_unskip_unkern:
3566         \__enumext_unskip_unkern:
3567         \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
3568       }
3569   }
```

(*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.

```
3570 \cs_new_protected:Nn \__enumext_after_list:
3571   {
3572     \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
3573       {
3574         \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3575           {
3576             \msg_warning:nn { enumext } { missing-miniright }
3577             \miniright
3578           }
3579         \int_gzero:N \g__enumext_minipage_stat_int
3580         \__enumext_unskip_unkern: % remove topsep + [partopsep]
3581         \end__enumext_mini_page
3582       }
3583       {
3584         \__enumext_multicols_stop:
3585       }
```

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.

```
3586     \__enumext_after_stop_list:
3587     \__enumext_check_ans_key_hook:
3588     \__enumext_vspace_below:
3589     \bool_set_false:N \l__enumext_standar_bool
```

```
3590        \__enumext_resume_save_counter:
3591      }
```

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.

```
3592 \__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.

```
3593 \NewDocumentEnvironment{keyans}{ O{} }
3594   {
3595      \__enumext_keyans_safe_exec:
3596      \__enumext_keyans_parse_keys:n {#1}
3597      \__enumext_before_list_v:
3598      \__enumext_start_list:nn
3599        { \tl_use:N \l__enumext_label_v_tl }
3600        {
3601           \__enumext_list_arg_two_v:
3602           \__enumext_before_keys_exec_v:
3603        }
3604      \__enumext_keyans_set_item_width:
3605      \__enumext_after_args_exec_v:
3606   }
3607   {
3608      \__enumext_check_starred_cmd:n { item }
3609      \__enumext_stop_list:
3610      \__enumext_after_list_v:
3611   }
```

(*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.

```
3612 \cs_new_protected:Nn \__enumext_keyans_set_item_width:
3613   {
3614      \dim_set:Nn \itemwidth
3615        {
3616           \linewidth
3617        }
3618      \dim_compare:nT
3619        {
3620           \l__enumext_listoffset_v_dim != \c_zero_dim
3621        }
3622        {
3623           \dim_sub:Nn \itemwidth
3624             {
3625                \l__enumext_listoffset_v_dim
3626             }
3627        }
3628   }
```

(*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 Implement. 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.

```
3629 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3630   {
3631      \bool_if:NF \l__enumext_store_active_bool
3632        {
3633           \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
3634        }
3635      \int_incr:N \l__enumext_keyans_level_int
3636      \bool_set_true:N \l__enumext_keyans_env_bool
3637      \__enumext_keyans_name_and_start:
```

```
3638        % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3639        \bool_set_false:N \l__enumext_store_active_bool
3640        \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
3641          {
3642            \msg_error:nn { enumext } { keyans-nested }
3643          }
3644        \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3645          {
3646            \msg_error:nn { enumext } { keyans-wrong-level }
3647          }
3648      }
```

(*End of definition for* __enumext_keyans_safe_exec:.)

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

```
3649  \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3650    {
3651      \keys_set:nn { enumext / keyans } {#1}
3652    }
```

(*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:

```
3653  \cs_new_protected:Nn \__enumext_before_list_v:
3654    {
3655      \__enumext_vspace_above_v:
3656      \__enumext_before_args_exec_v:
3657      \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3658        {
3659          \dim_set:Nn \l__enumext_minipage_left_v_dim
3660            {
3661              \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3662            }
3663          \bool_set_true:N \l__enumext_minipage_active_v_bool
3664          \int_gincr:N \g__enumext_minipage_stat_int
3665          \__enumext_keyans_minipage_add_space:
3666          \__enumext_mini_page{ \l__enumext_minipage_left_v_dim }
3667        }
3668      \__enumext_keyans_multicols_start:
3669    }
3670  \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3671    {
3672      \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3673        {
3674          \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3675            {
3676              \dim_set:Nn \l__enumext_columns_sep_v_dim
3677                {
3678                  (
3679                    \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3680                  ) / \l__enumext_columns_v_int
3681                  - \l__enumext_listoffset_v_dim
3682                }
3683            }
3684          \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim
3685          \dim_zero:N \columnseprule % no rule here
3686          \bool_if:NF \l__enumext_minipage_active_v_bool
3687            {
3688              \skip_zero:N \multicolsep
3689              \__enumext_keyans_multi_addvspace:
3690            }
3691          \raggedcolumns
3692          \begin{multicols}{ \l__enumext_columns_v_int }
3693        }
3694    }
3695  \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3696    {
3697      \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3698        {
3699          \end{multicols}
3700          \__enumext_unskip_unkern:
```

```
3701            \__enumext_unskip_unkern:
3702            \par\addvspace{ \l__enumext_multicols_below_v_skip }
3703          }
3704      }
3705  \cs_new_protected:Nn \__enumext_after_list_v:
3706      {
3707        \bool_if:NTF \l__enumext_minipage_active_v_bool
3708          {
3709            \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3710              {
3711                \msg_warning:nn { enumext } { missing-miniright }
3712                \miniright
3713              }
3714            \int_gzero:N \g__enumext_minipage_stat_int
3715            \__enumext_unskip_unkern: % remove topsep + [partopsep]
3716            \end__enumext_mini_page
3717            \par\addvspace{ \l__enumext_minipage_after_skip }
3718          }
3719          {
3720            \__enumext_keyans_multicols_stop:
3721          }
3722        \bool_set_false:N \l__enumext_keyans_env_bool
3723        \__enumext_after_stop_list_v:
3724        \__enumext_vspace_below_v:
3725      }
```

(*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*

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

<div style="text-align:right">
start-list-tags
stop-start-tags
stop-list-tags
__enumext_start_list_tag:n
__enumext_stop_start_list_tag:
__enumext_stop_list_tag:n
</div>

```
3726  \socket_new:nn {tagsupport/enumext/starred}{ 1 }
3727  \socket_new_plug:nnn {tagsupport/enumext/starred} {start-list-tags}
3728    {
3729      \ResumeTagging{#1}
3730      \tag_struct_begin:n {tag=LI}
3731      \tag_struct_begin:n {tag=Lbl}
3732      \tag_mc_begin:n {tag=Lbl}
3733    }
3734  \socket_new_plug:nnn {tagsupport/enumext/starred} {stop-start-tags}
3735    {
3736      \tag_mc_end:
3737      \tag_struct_end:n {tag=Lbl}
3738      \tag_struct_begin:n {tag=LBody}
3739      \tag_struct_begin:n {tag=text-unit}
3740      \tag_struct_begin:n {tag=text}
3741    }
3742  \socket_new_plug:nnn {tagsupport/enumext/starred} {stop-list-tags}
3743    {
3744      \tag_struct_end:n {tag=text}
3745      \tag_struct_end:n {tag=text-unit}
3746      \tag_struct_end:n {tag=LBody}
3747      \tag_struct_end:n {tag=LI}
3748      \SuspendTagging{#1}
3749    }
```

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

```
3750  \cs_new_protected_nopar:Npn \__enumext_start_list_tag:n #1
3751    {
3752      \IfDocumentMetadataTF
3753        {
3754          \socket_assign_plug:nn {tagsupport/enumext/starred} {start-list-tags}
```

```
3755          \socket_use:n {tagsupport/enumext/starred} {#1}
3756        }
3757      {}
3758    }
3759 \cs_new_protected_nopar:Nn \__enumext_stop_start_list_tag:
3760    {
3761      \IfDocumentMetadataTF
3762        {
3763          \socket_assign_plug:nn {tagsupport/enumext/starred} {stop-start-tags}
3764          \socket_use:nn {tagsupport/enumext/starred} { }
3765        }
3766      {}
3767    }
3768 \cs_new_protected_nopar:Npn \__enumext_stop_list_tag:n #1
3769    {
3770      \IfDocumentMetadataTF
3771        {
3772          \socket_assign_plug:nn {tagsupport/enumext/starred} {stop-list-tags}
3773          \socket_use:nn {tagsupport/enumext/starred} {#1}
3774        }
3775      {}
3776    }
```

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

12.40.2   Socket for tagging support in keyanspic

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

```
3777 \socket_new:nn {tagsupport/enumext/keyanspic}{ 0 }
3778 \socket_new_plug:nnn {tagsupport/enumext/keyanspic} {start-list-tags}
3779    {
3780      \ResumeTagging{keyanspic}
3781      \tag_start:n {keyanspic}
3782      \tag_struct_begin:n {tag=LI}
3783      \tag_struct_begin:n {tag=Lbl}
3784      \tag_mc_begin:n {tag=Lbl}
3785    }
3786 \socket_new_plug:nnn {tagsupport/enumext/keyanspic} {stop-start-tags}
3787    {
3788      \tag_mc_end:
3789      \tag_struct_end:n {tag=Lbl}
3790      \tag_struct_begin:n {tag=LBody}
3791      \tag_struct_begin:n {tag=text-unit}
3792      \tag_struct_begin:n {tag=text}
3793      \tag_mc_begin:n {tag=text}
3794    }
3795 \socket_new_plug:nnn {tagsupport/enumext/keyanspic} {stop-list-tags}
3796    {
3797      \tag_mc_end:
3798      \tag_struct_end:n {tag=text-unit}
3799      \tag_struct_end:n {tag=text}
3800      \tag_struct_end:n {tag=LBody}
3801      \tag_struct_end:n {tag=LI}
3802      \tag_stop:n {keyanspic}
3803      \SuspendTagging{keyanspic}
3804    }
```

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

```
3805 \cs_new_protected_nopar:Nn \__enumext_anspic_start_list_tag:
3806    {
3807      \IfDocumentMetadataTF
3808        {
3809          \socket_assign_plug:nn {tagsupport/enumext/keyanspic} {start-list-tags}
3810          \socket_use:n {tagsupport/enumext/keyanspic}
3811        }
3812      {}
3813    }
3814 \cs_new_protected_nopar:Nn \__enumext_anspic_stop_start_list_tag:
3815    {
3816      \IfDocumentMetadataTF
3817        {
3818          \socket_assign_plug:nn {tagsupport/enumext/keyanspic} {stop-start-tags}
3819          \socket_use:nn {tagsupport/enumext/keyanspic}
```

```
3820          }
3821        {}
3822    }
3823 \cs_new_protected_nopar:Nn \__enumext_anspic_stop_list_tag:
3824    {
3825      \IfDocumentMetadataTF
3826        {
3827          \socket_assign_plug:nn {tagsupport/enumext/keyanspic} {stop-list-tags}
3828          \socket_use:nn {tagsupport/enumext/keyanspic}
3829        }
3830        {}
3831    }
```

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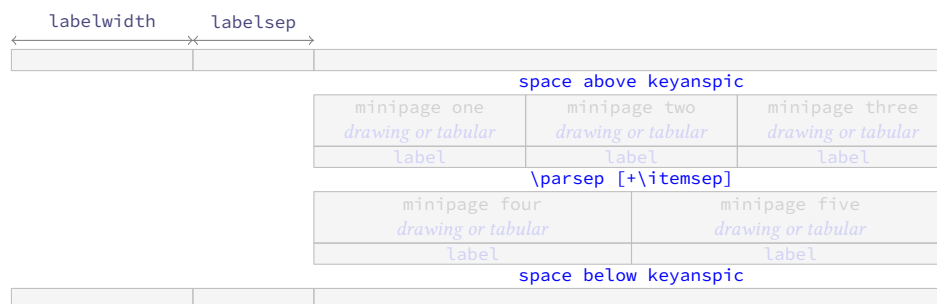
Figure 12: Representation of the `keyanspic` spacing in enumext.

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 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.

```
3832 \NewDocumentCommand \anspic { s o +m }
3833    {
```

We check that the command is active in the `keyanspic` environment only if the `save-ans` key is present, otherwise we return an error.

```
3834      \bool_if:NF \l__enumext_store_active_bool
3835        {
3836          \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
3837        }
3838      \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3839        {
3840          \msg_error:nn { enumext } { keyanspic-wrong-level }
3841        }
3842      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
3843        {
3844          \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
3845        }
```

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.

```
3846      \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
3847        {
3848          \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
3849        }
3850    }
```

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

__enumext_anspic_box_set_dim:n
__enumext_keyans_anspic_label:nnn
__enumext_keyans_anspic_code:nnn

The function __enumext_keyans_anspic_code:nnn will be in charge of handling the *"counter"* and ⟨*label*⟩, which will have the same configuration as the keyans environment.

```
3851 \str_new:N \l__enumext_keyans_pic_label_pos_str
3852 \bool_new:N \l__enumext_keyans_pic_star_bool
3853 \box_new:N \l__enumext_anspic_label_box
3854 \box_new:N \l__enumext_anspic_body_box
3855 \dim_new:N \l__enumext_anspic_label_htdp_dim
3856 \dim_new:N \l__enumext_anspic_body_htdp_dim
3857 % box for dim
3858 \cs_new_protected:Npn \__enumext_anspic_box_set_dim:n #1
3859   {
3860     % body
3861     \vbox_set:Nn \l__enumext_anspic_body_box { #1 }
3862     \dim_set:Nn \l__enumext_anspic_body_htdp_dim
3863       {
3864         \box_ht_plus_dp:N \l__enumext_anspic_body_box
3865       }
3866     % label
3867     \vbox_set:Nn \l__enumext_anspic_label_box { \l__enumext_label_v_tl }
3868     \dim_set:Nn \l__enumext_anspic_label_htdp_dim
3869       {
3870         \box_ht_plus_dp:N \l__enumext_anspic_label_box
3871       }
3872   }
3873 % process label
3874 \cs_new_protected:Npn \__enumext_anspic_label:nn #1 #2
3875   {
3876     \bool_if:nT { #1 }
3877       {
3878         \__enumext_keyans_addto_prop:n { #2 }
3879         \__enumext_keyans_store_ref:
3880         \__enumext_keyans_addto_seq:n { #2 }
3881         \int_gincr:N \g__enumext_check_starred_cmd_int
3882         \bool_lazy_or:nnT
3883           { \bool_if_p:N \l__enumext_show_answer_bool }
3884           { \bool_if_p:N \l__enumext_show_position_bool }
3885           {
3886             \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
3887             \__enumext_keyans_show_left:n { #2 }
3888             \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
3889           }
3890       }
3891     \makebox[ \l__enumext_keyans_pic_width_dim ][ c ]
3892       {
3893         \tl_use:N \l__enumext_label_font_style_v_tl
3894         \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
3895       }
3896   }
3897 \cs_new_protected:Npn \__enumext_keyans_anspic_label:nnn #1 #2 #3
3898   {
3899     \stepcounter { enumXvi }
3900     \__enumext_anspic_box_set_dim:n { #3 }
3901     \bool_if:NTF \l__enumext_keyans_pic_star_bool
3902       {
3903         \__enumext_anspic_label:nn { #1 } { #2 }
3904       }
3905       {
3906         \raisebox
3907           {
3908             -\dim_eval:n
3909               {
3910                 \l__enumext_anspic_label_htdp_dim
3911                 + \l__enumext_anspic_body_htdp_dim
3912                 + \box_ht_plus_dp:N \strutbox
3913               }
3914           }
3915         [ 0pt ] [ 0pt ]
3916           {
3917             \__enumext_anspic_label:nn { #1 } { #2 }
3918           }
3919       }
```

```
3920      }
3921  \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
3922    {
3923      \__enumext_anspic_start_list_tag:
3924      \__enumext_keyans_anspic_label:nnn { #1 } { #2 } { #3 }
3925      \__enumext_anspic_stop_start_list_tag:
3926      \\ #3
3927      \__enumext_anspic_stop_list_tag:
3928    }
```

(*End of definition for* __enumext_anspic_box_set_dim:n*,* __enumext_keyans_anspic_label:nnn*, and* __enumext_keyans_-
anspic_code:nnn*.*)

12.41.2   The environment keyanspic

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.

```
3929  \NewDocumentEnvironment{keyanspic}{ s o }
3930    {
3931      \__enumext_keyans_pic_safe_exec:
3932      \bool_if:nTF { #1 }
3933        {
3934          \bool_set_true:N \l__enumext_keyans_pic_star_bool
3935          \str_set:Nn \l__enumext_keyans_pic_label_pos_str { t }
3936        }
3937        {
3938          \str_set:Nn \l__enumext_keyans_pic_label_pos_str { b }
3939        }
3940      \begin{list} { } { \__enumext_keyans_pic_arg_two: }
3941      \SuspendTagging{list} % non op in LaTeX2e
3942      \item[] \scan_stop:
3943      % paranoia
3944      \RenewDocumentCommand \item {}
3945        {
3946          \msg_error:nn { enumext } { keyanspic-item-cmd }
3947        }
3948      \IfDocumentMetadataTF
3949        {
3950          %\tagpdfparaOff (not work see chat TSX)
3951          \ResumeTagging{keyanspic}
3952          \tag_tool:n {para/tagging=false}
3953          \SuspendTagging{keyanspic} % non op in LaTeX2e
3954        } { }
3955      \vspace { \l__enumext_keyans_pic_above_skip } % need see what this on space
3956    }
```

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.

```
3957    {
3958      \IfDocumentMetadataTF
3959        {
3960          \tag_start:n {keyanspic}
3961          \tag_struct_begin:n {tag=L,attribute=enumerate}
3962        }{ }
3963      \tl_if_novalue:nTF { #2 }
3964        {
3965          \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
3966        }
3967        { \__enumext_keyans_pic_do:n { #2 } }
3968      \IfDocumentMetadataTF { \tag_stop:n {keyanspic} } { }
3969      \end{list}
3970      \IfDocumentMetadataTF { \tag_struct_end: \tag_struct_end: } { }
3971      \__enumext_check_starred_cmd:n { anspic }
3972      \setcounter { enumXvi } { 0 }
3973      %%\vspace { \l__enumext_topsep_v_skip }
3974      \par\addvspace{ \dim_eval:n { -\parsep + \box_dp:N \strutbox } }
3975      %\bool_set_false:N \l__enumext_store_active_bool
3976    }
```

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

__enumext_keyans_pic_safe_exec:

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

```
3977 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
3978   {
3979     \int_incr:N \l__enumext_keyans_pic_level_int
3980     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
3981       {
3982         \msg_error:nn { enumext } { keyanspic-nested }
3983       }
3984     \__enumext_keyans_name_and_start:
3985   }
```

(*End of definition for* __enumext_keyans_pic_safe_exec:.)

__enumext_keyans_pic_skip_abs:N

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

```
3986 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3987   {
3988     \dim_compare:nNnT { #1 } < { 0pt }
3989       { \skip_set:Nn #1 { -#1 } }
3990   }
```

(*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.

```
3991 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
3992   {
```

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.

```
3993       \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3994       \__enumext_list_arg_two_v:
```

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.

```
3995       %%%\skip_add:Nn \parsep { \itemsep }
3996       \dim_add:Nn  \leftmargin { -\labelwidth - \labelsep }
3997       \dim_zero:N  \labelwidth
3998       \dim_zero:N  \listparindent
3999       \dim_zero:N  \labelsep
4000       \skip_zero:N \partopsep
4001       \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.

```
4002       \__enumext_keyans_pic_skip_abs:N \parsep
4003       \skip_set:Nn \l__enumext_keyans_pic_above_skip
4004         {
4005           \box_dp:N \strutbox
4006           + \l__enumext_topsep_v_skip
4007           %%%- \parsep
4008         }
4009   }
```

(*End of definition for* __enumext_keyans_pic_arg_two:.)

__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.

```
4010 \cs_new_protected:Nn \__enumext_keyans_pic_do:n
4011   {
4012     \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
4013   }
4014 \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.

```
4015 \cs_new_protected:Nn \__enumext_keyans_pic_row:n
4016   {
4017     \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
4018     \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
4019     \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
4020     \int_step_inline:nnn
4021       { \l__enumext_keyans_pic_above_int + 1 }
4022       { \l__enumext_keyans_pic_below_int }
4023       {
4024         \IfDocumentMetadataTF { \tag_stop:n {minipage} } { }
4025         \begin{minipage}[ \l__enumext_keyans_pic_label_pos_str ]{ \l__enumext_keyans_pic_width_dim
4026           \centering
4027           \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
4028         \end{minipage}
4029         \IfDocumentMetadataTF { \tag_start:n {minipage} } { }
4030       }
4031     \bool_if:NTF \l__enumext_keyans_pic_star_bool
4032       {
4033         \par
4034       }
4035       {
4036         \par%\vspace{ \box_ht_plus_dp:N \strutbox }
4037       }
4038   }
```

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

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.

```
4039 \cs_new_protected:Nn \__enumext_footnotetext:nn
4040   {
4041     \footnotetext[#1]{#2}
4042   }
4043 \cs_new_protected:Nn \__enumext_renew_footnote:
4044   {
4045     \seq_gclear:N \g__enumext_footnote_arg_seq
4046     \seq_gclear:N \g__enumext_footnote_int_seq
4047     \RenewDocumentCommand \footnote { o +m }
4048       {
4049         \tl_if_novalue:nTF {##1}
4050           {
4051             \stepcounter{footnote}
4052             \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
4053           }
4054           {
4055             \int_gset:Nn \g__enumext_footnote_int { ##1 }
4056           }
4057         \footnotemark [ \g__enumext_footnote_int ]
4058         \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
4059         \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
4060       }
4061   }
4062 \cs_new_protected:Nn \__enumext_print_footnote:
4063   {
4064     \seq_if_empty:NF \g__enumext_footnote_int_seq
4065       {
4066         \seq_map_pairwise_function:NNN
4067           \g__enumext_footnote_int_seq
4068           \g__enumext_footnote_arg_seq
4069           \__enumext_footnotetext:nn
```

```
4070              }
4071         }
```

(*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.

```
4072 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
4073     {
4074         \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
4075             {
4076                 \dim_set:Nn \l__enumext_columns_sep_vii_dim
4077                     {
4078                         ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
4079                         / \l__enumext_columns_vii_int
4080                     }
4081             }
4082         \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - 1 }
4083         \dim_set:Nn \l__enumext_item_width_vii_dim
4084             {
4085                 ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
4086                 / \l__enumext_columns_vii_int
4087                 - \l__enumext_labelwidth_vii_dim
4088                 - \l__enumext_labelsep_vii_dim
4089             }
```

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

```
4090         \dim_compare:nNnT { \l__enumext_rightmargin_vii_dim } > { \c_zero_dim }
4091             {
4092                 \dim_sub:Nn \l__enumext_item_width_vii_dim
4093                     {
4094                         ( \l__enumext_rightmargin_vii_dim * \l__enumext_tmpa_vii_int )
4095                         / \l__enumext_columns_vii_int
4096                     }
4097                 \dim_add:Nn \l__enumext_columns_sep_vii_dim
4098                     {
4099                         \l__enumext_rightmargin_vii_dim
4100                     }
4101             }
4102     }
```

Same implementation for the keyans* environment.

```
4103 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
4104     {
4105         \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
4106             {
4107                 \dim_set:Nn \l__enumext_columns_sep_viii_dim
4108                     {
4109                         ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
4110                         / \l__enumext_columns_viii_int
4111                     }
4112             }
4113         \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - 1 }
4114         \dim_set:Nn \l__enumext_item_width_viii_dim
4115             {
4116                 ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
4117                 / \l__enumext_columns_viii_int
4118                 - \l__enumext_labelwidth_viii_dim
4119                 - \l__enumext_labelsep_viii_dim
4120             }
4121         \dim_compare:nNnT { \l__enumext_rightmargin_viii_dim } > { \c_zero_dim }
4122             {
4123                 \dim_sub:Nn \l__enumext_item_width_viii_dim
4124                     {
4125                         ( \l__enumext_rightmargin_viii_dim * \l__enumext_tmpa_vii_int )
4126                         / \l__enumext_columns_viii_int
```

```
4127                    }
4128            \dim_add:Nn \l__enumext_columns_sep_viii_dim
4129                {
4130                    \l__enumext_rightmargin_viii_dim
4131                }
4132        }
4133    }
```

(*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.

```
4134  \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
4135    {
4136        \int_set:Nn \l__enumext_joined_item_vii_int {#1}
4137        \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
4138            {
4139                \msg_warning:nnee { enumext } { item-joined }
4140                    { \int_use:N \l__enumext_joined_item_vii_int }
4141                    { \int_use:N \l__enumext_columns_vii_int }
4142                \int_set:Nn \l__enumext_joined_item_vii_int
4143                    {
4144                        \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4145                    }
4146            }
4147        \int_compare:nNnT
4148            { \l__enumext_joined_item_vii_int }
4149                >
4150            { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4151            {
4152                \msg_warning:nnee { enumext } { item-joined-columns }
4153                    { \int_use:N \l__enumext_joined_item_vii_int }
4154                    {
4155                        \int_eval:n
4156                            { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4157                    }
4158                \int_set:Nn \l__enumext_joined_item_vii_int
4159                    {
4160                        \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4161                    }
4162            }
4163        \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { 1 }
4164            {
4165                \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
4166                \int_decr:N \l__enumext_joined_item_aux_vii_int
4167                \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
4168                \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
4169                \dim_set:Nn \l__enumext_joined_width_vii_dim
4170                    {
4171                        \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
4172                        + (  \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
4173                            + \l__enumext_columns_sep_vii_dim
4174                        )*\l__enumext_joined_item_aux_vii_int
4175                    }
4176                \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
4177            }
4178            {
4179                \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
4180                \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
4181            }
4182    }
```

Same implementation for the keyans* environment.

```
4183  \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
4184    {
4185        \int_set:Nn \l__enumext_joined_item_viii_int {#1}
4186        \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
4187            {
4188                \msg_warning:nnee { enumext } { item-joined }
```

```
4189          { \int_use:N \l__enumext_joined_item_viii_int }
4190          { \int_use:N \l__enumext_columns_viii_int }
4191        \int_set:Nn \l__enumext_joined_item_viii_int
4192          {
4193            \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4194          }
4195      }
4196    \int_compare:nNnT
4197      { \l__enumext_joined_item_viii_int }
4198        >
4199      { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4200      {
4201        \msg_warning:nnee { enumext } { item-joined-columns }
4202          { \int_use:N \l__enumext_joined_item_viii_int }
4203          {
4204            \int_eval:n
4205              { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4206          }
4207        \int_set:Nn \l__enumext_joined_item_viii_int
4208          {
4209            \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4210          }
4211      }
4212    \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { 1 }
4213      {
4214        \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
4215        \int_decr:N \l__enumext_joined_item_aux_viii_int
4216        \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
4217        \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
4218        \dim_set:Nn \l__enumext_joined_width_viii_dim
4219          {
4220            \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
4221          + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
4222            + \l__enumext_columns_sep_viii_dim
4223          )*\l__enumext_joined_item_aux_viii_int
4224          }
4225        \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
4226      }
4227      {
4228        \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
4229        \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
4230      }
4231    }
```

(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:`
`\__enumext_stop_mini_vii:`

The implementation of the `mini-env` key support is almost identical to the one used in the `enumext` and `keyans` environments, the difference is that the `__enumext_mini_page` environment on the *"right side"* is executed *"after"* closing the environment, so it is necessary to make a global copy of the variable `\l__enumext_minipage_right_vii_dim` in the variable `\g__enumext_minipage_right_vii_dim`.

```
4232  \cs_new_protected:Nn \__enumext_start_mini_vii:
4233    {
4234      \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
4235        {
4236          \dim_set:Nn \l__enumext_minipage_left_vii_dim
4237            {
4238              \linewidth
4239              - \l__enumext_minipage_right_vii_dim
4240              - \l__enumext_minipage_hsep_vii_dim
4241            }
4242          \bool_set_true:N \l__enumext_minipage_active_vii_bool
4243          \dim_gset_eq:NN
4244            \g__enumext_minipage_right_vii_dim
4245            \l__enumext_minipage_right_vii_dim
4246          \__enumext_mini_addvspace_vii:
4247          \nointerlineskip\noindent
4248          \__enumext_mini_page{ \l__enumext_minipage_left_vii_dim }
4249        }
4250    }
```

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"*.

```
4251 \cs_new_protected:Nn \__enumext_stop_mini_vii:
4252   {
4253     \bool_if:NT \l__enumext_minipage_active_vii_bool
4254       {
4255         \end__enumext_mini_page
4256         \hfill
4257         \bool_gset_true:N \g__enumext_minipage_active_vii_bool
4258       }
4259   }
```

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.

```
4260 \__enumext_after_env:nn {enumext*}
4261   {
4262     \bool_if:NT \g__enumext_minipage_active_vii_bool
4263       {
4264         \__enumext_mini_page{ \g__enumext_minipage_right_vii_dim }
4265           \par\addvspace { \g__enumext_minipage_right_skip }
4266           \bool_if:NF \g__enumext_minipage_center_vii_bool
4267             {
4268               \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
4269                 {
4270                   \centering
4271                 }
4272             }
4273           \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
4274             {
4275               \tl_use:N \g__enumext_miniright_code_vii_tl
4276             }
4277           \box_use_drop:N \l__enumext_miniright_code_vii_box
4278         \end__enumext_mini_page
4279         \par\addvspace{ \g__enumext_minipage_after_skip }
4280       }
4281     \bool_gset_false:N \g__enumext_minipage_active_vii_bool
4282     \bool_gset_true:N \g__enumext_minipage_center_vii_bool
4283     \tl_gclear:N \g__enumext_miniright_code_vii_tl
4284     \dim_gzero:N \g__enumext_minipage_right_vii_dim
4285     \bool_gset_false:N \g__enumext_starred_bool
4286   }
```

(*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.

```
4287 \cs_new_protected:Nn \__enumext_start_mini_viii:
4288   {
4289     \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
4290       {
4291         \dim_set:Nn \l__enumext_minipage_left_viii_dim
4292           {
4293             \linewidth
4294             - \l__enumext_minipage_right_viii_dim
4295             - \l__enumext_minipage_hsep_viii_dim
4296           }
4297         \bool_set_true:N \l__enumext_minipage_active_viii_bool
4298         \dim_gset_eq:NN
4299           \g__enumext_minipage_right_viii_dim
4300           \l__enumext_minipage_right_viii_dim
4301         \__enumext_mini_addvspace_viii:
4302         \nointerlineskip\noindent
4303         \__enumext_mini_page{ \l__enumext_minipage_left_viii_dim }
4304       }
4305   }
4306 \cs_new_protected:Nn \__enumext_stop_mini_viii:
4307   {
```

```
4308      \bool_if:NT \l__enumext_minipage_active_viii_bool
4309        {
4310          \end__enumext_mini_page
4311          \hfill
4312          \bool_gset_true:N \g__enumext_minipage_active_viii_bool
4313        }
4314    }
4315 \__enumext_after_env:nn {keyans*}
4316    {
4317      \bool_if:NT \g__enumext_minipage_active_viii_bool
4318        {
4319          \__enumext_mini_page{ \g__enumext_minipage_right_viii_dim }
4320            \par\addvspace { \g__enumext_minipage_right_skip }
4321            \bool_if:NF \g__enumext_minipage_center_viii_bool
4322              {
4323                \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
4324                  {
4325                    \centering
4326                  }
4327              }
4328            \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
4329              {
4330                \tl_use:N \g__enumext_miniright_code_viii_tl
4331              }
4332            \box_use_drop:N \l__enumext_miniright_code_viii_box
4333          \end__enumext_mini_page
4334          \par\addvspace{ \g__enumext_minipage_after_skip }
4335        }
4336      \bool_gset_false:N \g__enumext_minipage_active_viii_bool
4337      \bool_gset_true:N \g__enumext_minipage_center_viii_bool
4338      \tl_gclear:N \g__enumext_miniright_code_viii_tl
4339      \dim_gzero:N \g__enumext_minipage_right_viii_dim
4340    }
```

(*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.

```
4341 \NewDocumentEnvironment{enumext*}{ o }
4342    {
4343      \__enumext_safe_exec_vii:
4344      \__enumext_parse_keys_vii:n {#1}
4345      \__enumext_before_list_vii:
4346      \__enumext_start_store_level_vii:
4347      \__enumext_start_list:nn { }
4348        {
4349          \__enumext_list_arg_two_vii:
4350          \__enumext_before_keys_exec_vii:
4351        }
4352      % Stop tagging
4353      \SuspendTagging{enumext*}
4354      \__enumext_starred_columns_set_vii:
4355      \item[] \scan_stop:
4356      \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_first_item_tmp_vii:
4357      \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
4358      \ignorespaces
4359    }
4360    {
4361      % Close for first \item
4362      \IfDocumentMetadataTF { \tag_struct_end: } { }
4363      \__enumext_stop_item_tmp_vii:
4364      \__enumext_remove_extra_parsep_vii:
4365      \__enumext_stop_list:
4366      \__enumext_stop_store_level_vii:
4367      \__enumext_after_list_vii:
4368    }
```

(*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.

```
4369 \cs_new_protected:Nn \__enumext_safe_exec_vii:
4370   {
4371     \__enumext_internal_mini_page:
4372     \__enumext_is_not_nested:
4373     \int_incr:N \l__enumext_level_h_int
4374     \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
4375       {
4376         \msg_error:nn { enumext } { nested }
4377       }
4378     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
4379       {
4380         \msg_error:nnn { enumext } { nested-horizontal } { keyans*}
4381       }
4382     \bool_set_true:N \l__enumext_starred_bool
4383     \bool_set_false:N \l__enumext_standar_bool
4384     \__enumext_is_on_first_level:
4385   }
```

(*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.

```
4386 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
4387   {
4388     \tl_if_novalue:nF {#1}
4389       {
4390         \str_clear:N \l__enumext_series_str
4391         \keys_set:nn { enumext / enumext* } {#1}
4392         \__enumext_parse_series:n {#1}
4393         \__enumext_store_active_keys_vii:n {#1}
4394         \__enumext_nested_base_line_fix:
4395       }
4396   }
```

(*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*.

```
4397 \cs_new_protected:Nn \__enumext_before_list_vii:
4398   {
4399     \__enumext_vspace_above_vii:
4400     \__enumext_check_ans_active:
4401     \__enumext_before_args_exec_vii:
4402     \__enumext_start_mini_vii:
4403   }
```

(*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.

```
4404 \cs_new_protected:Nn \__enumext_after_list_vii:
4405   {
```

```
4406        \__enumext_stop_mini_vii:
4407        \__enumext_after_stop_list_vii:
4408        \__enumext_check_ans_key_hook:
4409        \__enumext_vspace_below_vii:
4410        \bool_set_false:N \l__enumext_starred_bool
4411        \__enumext_resume_save_counter:
4412      }
```

(*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.

```
4413  \cs_new_protected:Nn \__enumext_start_store_level_vii:
4414    {
4415      \bool_if:NT \l__enumext_store_active_bool
4416        {
4417          \int_compare:nNnT { \l__enumext_level_int } > { 0 }
4418            {
4419              \__enumext_store_level_open_vii:
4420            }
4421        }
4422    }
4423  \cs_new_protected:Nn \__enumext_stop_store_level_vii:
4424    {
4425      \bool_if:NT \l__enumext_store_active_bool
4426        {
4427          \int_compare:nNnT { \l__enumext_level_int } > { 0 }
4428            {
4429              \__enumext_store_level_close_vii:
4430            }
4431        }
4432    }
```

(*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.

```
4433  \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_vii:
4434    {
4435      \skip_horizontal:n { -\l__enumext_labelwidth_vii_dim - \l__enumext_labelsep_vii_dim }
4436    }
```

(*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 incre-
ment 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.

```
4437  \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
4438    {
4439      \__enumext_stop_item_tmp_vii:
4440      \int_incr:N \l__enumext_item_column_pos_vii_int
4441      \int_gincr:N \g__enumext_item_count_all_vii_int
4442      \__enumext_item_peek_args_vii:
4443    }
```

(*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 argu-
ment "(", 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).

```
4444  \cs_new_protected:Nn \__enumext_item_peek_args_vii:
4445    {
4446      \peek_meaning:NTF (
4447        { \__enumext_joined_item_vii:w }
4448        { \__enumext_joined_item_vii:w (1) }
4449    }
```

(*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.

```
4450 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
4451   {
4452     \__enumext_starred_joined_item_vii:n {#1}
4453     \peek_meaning_remove:NTF *
4454       { \__enumext_starred_item_vii:w  }
4455       { \__enumext_standar_item_vii:w }
4456   }
```

(*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].

```
4457 \cs_new_protected:Npn \__enumext_standar_item_vii:w
4458   {
4459     \bool_set_false:N \l__enumext_item_starred_vii_bool
4460     \peek_meaning:NTF [
4461       {
4462         \bool_set_eq:NN \l__enumext_wrap_label_vii_bool \l__enumext_wrap_label_opt_vii_bool
4463         \__enumext_start_item_vii:w
4464       }
4465       {
4466         \bool_set_true:N \l__enumext_wrap_label_vii_bool
4467         \legacy_if_set_true:n { @noitemarg }
4468         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4469       }
4470   }
```

(*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*⟩].

```
4471 \cs_new_protected:Npn \__enumext_starred_item_vii:w
4472   {
4473     \bool_set_true:N \l__enumext_item_starred_vii_bool
4474     \bool_set_true:N \l__enumext_wrap_label_vii_bool
4475     \peek_meaning:NTF [
4476       { \__enumext_starred_item_vii_aux_i:w }
4477       { \__enumext_starred_item_vii_aux_ii:w }
4478   }
4479 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
4480   {
4481     \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
4482     \__enumext_starred_item_vii_aux_ii:w
4483   }
4484 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
4485   {
4486     \peek_meaning:NTF [
4487       { \__enumext_starred_item_vii_aux_iii:w }
4488       {
4489         \dim_set_eq:NN \l__enumext_item_symbol_sep_vii_dim \l__enumext_labelsep_vii_dim
4490         \legacy_if_set_true:n { @noitemarg }
4491         \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4492       }
4493   }
4494 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
4495   {
4496     \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
4497     \legacy_if_set_true:n { @noitemarg }
4498     \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4499   }
```

(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*.

```
4500  \cs_new_protected_nopar:Npn \__enumext_fake_make_label_vii:n #1
4501    {
4502      \legacy_if:nT { @noitemarg }
4503        {
4504          \legacy_if_set_false:n { @noitemarg }
4505          \legacy_if:nT { @nmbrlist }
4506            {
4507              \refstepcounter{enumXvii}
4508              \bool_if:NT \l__enumext_check_answers_bool
4509                {
4510                  \int_gincr:N \g__enumext_item_number_int
4511                  \bool_set_true:N \l__enumext_item_number_bool
4512                }
4513            }
4514        }
4515      \bool_if:NT \l__enumext_item_starred_vii_bool
4516        {
4517          \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
4518            {
4519              \tl_gset_eq:NN
4520                \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
4521            }
4522          \mode_leave_vertical:
4523          \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
4524          \hbox_overlap_left:n { \g__enumext_item_symbol_aux_vii_tl }
4525          \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
4526          \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
4527        }
4528      \bool_if:NTF \l__enumext_wrap_label_vii_bool
4529        {
4530          \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
4531            {
4532              \tl_use:N \l__enumext_label_font_style_vii_tl
4533                \__enumext_wrapper_label_vii:n {#1}
4534            }
4535        }
4536        {
4537          \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
4538            {
4539              \tl_use:N \l__enumext_label_font_style_vii_tl #1
4540            }
4541        }
4542      \skip_horizontal:N \l__enumext_labelsep_vii_dim
4543    }
```

(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.

```
4544  \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
4545    {
4546      \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
4547      \hbox_set_to_wd:Nnw \l__enumext_item_text_vii_box
4548        {
4549          \l__enumext_joined_width_vii_dim
```

```
4550          +  \l__enumext_labelwidth_vii_dim
4551          +  \l__enumext_labelsep_vii_dim
4552        }
```

If \DocumentMetadata is not active and the state of the variable \l__enumext_footnotes_key_bool is false, we will redefine the \footnote command.

```
4553      \IfDocumentMetadataTF { }
4554        {
4555          \bool_if:NF \l__enumext_footnotes_key_bool
4556            {
4557              \__enumext_renew_footnote:
4558            }
4559        }
```

Now we insert our *sockets* for the tagPDF support and print \item.

```
4560        \__enumext_start_list_tag:n {enumext*}
4561        \__enumext_fake_make_label_vii:n {#1}
4562        \__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.

```
4563        \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim  }
4564        \tl_use:N \l__enumext_after_list_args_vii_tl
4565        \dim_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
4566        \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
4567        \tl_use:N \l__enumext_fake_item_indent_vii_tl
4568      }
```

(*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*.

```
4569  \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
4570    {
4571        \__enumext_endminipage:
4572      \__enumext_stop_list_tag:n {enumext*}
4573      \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.

```
4574        \int_set:Nn \hbadness { 10000 }
4575        \box_use_drop:N \l__enumext_item_text_vii_box
4576        \IfDocumentMetadataTF { }
4577          {
4578            \bool_if:NF \l__enumext_footnotes_key_bool
4579              {
4580                \__enumext_print_footnote:
4581              }
4582          }
```

Finally set the *vertical* and *horizontal* spaces between rows and columns.

```
4583        \int_compare:nNnTF
4584          { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
4585          {
4586            \par\noindent
4587            \int_zero:N \l__enumext_item_column_pos_vii_int
4588          }
4589          {
4590            \skip_horizontal:N \l__enumext_columns_sep_vii_dim
4591          }
4592    }
```

(*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*⟩.

```
4593  \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
4594    {
4595        \int_compare:nNnT
```

```
4596       {
4597         \int_mod:nn
4598           {  \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
4599       }
4600     =
4601     { 0 }
4602     {
4603       \para_end:
4604       \skip_vertical:n { -\l__enumext_itemsep_vii_skip }
4605       \skip_vertical:N \c_zero_skip
4606       \int_gzero:N \g__enumext_item_count_all_vii_int
4607     }
4608   }
```

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.

```
4609 \__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.

```
4610 \NewDocumentEnvironment{keyans*}{ o }
4611   {
4612     \__enumext_safe_exec_viii:
4613     \__enumext_parse_keys_viii:n {#1}
4614     \__enumext_before_list_viii:
4615     \__enumext_start_list:nn { }
4616       {
4617         \__enumext_list_arg_two_viii:
4618         \__enumext_before_keys_exec_viii:
4619       }
4620     \__enumext_starred_columns_set_viii:
4621     \item[] \scan_stop:
4622     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_first_item_tmp_viii:
4623     \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
4624     \ignorespaces
4625   }
4626   {
4627     \__enumext_stop_item_tmp_viii:
4628     \__enumext_remove_extra_parsep_viii:
4629     \__enumext_check_starred_cmd:n { item }
4630     \__enumext_stop_list:
4631     \__enumext_after_list_viii:
4632   }
```

(*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.

```
4633 \cs_new_protected:Nn \__enumext_safe_exec_viii:
4634   {
4635     \bool_if:NF \l__enumext_store_active_bool
4636       {
4637         \msg_error:nnnn { enumext } { wrong-place }{ keyans* }{ save-ans }
4638       }
4639     \int_incr:N \l__enumext_keyans_level_h_int
4640     \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
4641       {
4642         \msg_error:nn { enumext } { nested }
4643       }
```

```
4644        \__enumext_keyans_name_and_start:
4645        \bool_if:NT \l__enumext_starred_bool
4646          {
4647            \msg_error:nnn { enumext } { nested-horizontal } { enumext* }
4648          }
4649        \bool_set_true:N \l__enumext_starred_bool
4650        % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
4651        \bool_set_false:N \l__enumext_store_active_bool
4652        \int_compare:nNnT { \l__enumext_level_int } > { 1 }
4653          {
4654            \msg_error:nn { enumext } { keyans-wrong-level }
4655          }
4656      }
```

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

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

```
4657  \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
4658    {
4659      \tl_if_novalue:nF {#1}
4660        {
4661          \keys_set:nn { enumext / keyans* } {#1}
4662        }
4663    }
```

(*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.

```
4664  \cs_new_protected:Nn \__enumext_before_list_viii:
4665    {
4666      \__enumext_vspace_above_viii:
4667      \__enumext_before_args_exec_viii:
4668      \__enumext_start_mini_viii:
4669    }
```

(*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.

```
4670  \cs_new_protected:Nn \__enumext_after_list_viii:
4671    {
4672      \__enumext_stop_mini_viii:
4673      \__enumext_after_stop_list_viii:
4674      \__enumext_vspace_below_viii:
4675    }
```

(*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.

```
4676  \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_viii:
4677    {
4678      \skip_horizontal:n { -\l__enumext_labelwidth_viii_dim - \l__enumext_labelsep_viii_dim }
4679    }
```

(*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`.

```
4680 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
4681   {
4682     \__enumext_stop_item_tmp_viii:
4683     \int_incr:N \l__enumext_item_column_pos_viii_int
4684     \int_gincr:N \g__enumext_item_count_all_viii_int
4685     \__enumext_item_peek_args_viii:
4686   }
```

(*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`).

```
4687 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
4688   {
4689     \peek_meaning:NTF (
4690       { \__enumext_joined_item_viii:w }
4691       { \__enumext_joined_item_viii:w (1) }
4692   }
```

(*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`.

```
4693 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
4694   {
4695     \__enumext_starred_joined_item_viii:n {#1}
4696     \peek_meaning_remove:NTF *
4697       { \__enumext_starred_item_viii:w  }
4698       { \__enumext_standar_item_viii:w }
4699   }
```

(*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 ]`.

```
4700 \cs_new_protected:Npn \__enumext_standar_item_viii:w
4701   {
4702     \bool_set_false:N \l__enumext_item_starred_viii_bool
4703     \peek_meaning:NTF [
4704       {
4705         \bool_set_eq:NN \l__enumext_wrap_label_viii_bool \l__enumext_wrap_label_opt_viii_bool
4706         \__enumext_start_item_viii:w
4707       }
4708       {
4709         \bool_set_true:N \l__enumext_wrap_label_viii_bool
4710         \legacy_if_set_true:n { @noitemarg }
4711         \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4712       }
4713   }
```

(*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*⟩`]`.

```
4714 \cs_new_protected:Npn \__enumext_starred_item_viii:w
4715   {
```

```
4716        \bool_set_true:N \l__enumext_item_starred_viii_bool
4717        \bool_set_true:N \l__enumext_wrap_label_viii_bool
4718        \peek_meaning:NTF [
4719          { \__enumext_starred_item_viii_aux_i:w }
4720          { \__enumext_starred_item_viii_aux_ii:w }
4721      }
```

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.

```
4722  \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
4723    {
4724      \tl_clear:N \l__enumext_store_current_label_tl
4725      \tl_if_novalue:nF { #1 }
4726        {
4727          \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
4728            {
4729              \tl_put_right:Ne \l__enumext_store_current_label_tl
4730                {
4731                  \l__enumext_store_keyans_item_opt_sep_tl
4732                }
4733              \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
4734            }
4735          \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
4736        }
4737      \__enumext_starred_item_viii_aux_ii:w
4738    }
4739  \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
4740    {
4741      \legacy_if_set_true:n { @noitemarg }
4742      \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4743    }
```

(*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.

```
4744  \cs_new_protected:Nn \__enumext_starred_item_exec:
4745    {
4746      \tl_put_left:Ne \l__enumext_store_current_label_tl { \l__enumext_label_viii_tl }
4747      \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
4748      \__enumext_keyans_store_ref:
4749      \tl_put_left:Ne \l__enumext_store_current_label_tl { \item }
4750      \__enumext_keyans_addto_seq_link:
4751      \int_gincr:N \g__enumext_check_starred_cmd_int
4752      \bool_if:NT \l__enumext_show_answer_bool
4753        {
4754          \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4755        }
4756      \bool_if:NT \l__enumext_show_position_bool
4757        {
4758          \tl_set:Ne \l__enumext_mark_answer_sym_tl
4759            {
4760              \group_begin:
4761                \exp_not:N \normalfont
4762                \exp_not:N \footnotesize [ \int_eval:n
4763                  {
4764                    \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
4765                  }
4766                ]
4767              \group_end:
4768            }
4769          \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4770        }
4771    }
```

(*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.

```
4772 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
4773   {
4774     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
4775     \legacy_if:nT { @noitemarg }
4776       {
4777         \legacy_if_set_false:n { @noitemarg }
4778         \legacy_if:nT { @nmbrlist }
4779           {
4780             \bool_if:NT \l__enumext_hyperref_bool
4781               {
4782                 \legacy_if_set_true:n { @hyper@item }
4783               }
4784             \refstepcounter{enumXviii}
4785           }
4786       }
```

Here we start capturing \item and its ⟨contents⟩ in a *horizontal box*.

```
4787     \hbox_set_to_wd:Nnw \l__enumext_item_text_viii_box
4788       {
4789         \l__enumext_joined_width_viii_dim
4790         + \l__enumext_labelwidth_viii_dim
4791         + \l__enumext_labelsep_viii_dim
4792       }
4793       \bool_if:NF \l__enumext_footnotes_key_bool
4794         {
4795           \__enumext_renew_footnote:
4796         }
4797       \bool_if:NT \l__enumext_item_starred_viii_bool
4798         {
4799           \__enumext_starred_item_exec:
4800         }
4801       \bool_if:NTF \l__enumext_wrap_label_viii_bool
4802         {
4803           \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
4804             {
4805               \tl_use:N \l__enumext_label_font_style_viii_tl
4806               \__enumext_wrapper_label_viii:n {#1}
4807             }
4808         }
4809         {
4810           \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
4811             {
4812               \tl_use:N \l__enumext_label_font_style_viii_tl #1
4813
4814             }
4815         }
4816       \skip_horizontal:N \l__enumext_labelsep_viii_dim
4817       \tl_use:N \l__enumext_after_list_args_viii_tl
4818       \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim  }
4819         \dim_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
4820         \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
4821         \bool_if:NT \l__enumext_item_starred_viii_bool
4822           {
4823             \tl_use:N \l__enumext_fake_item_indent_viii_tl
4824             \__enumext_keyans_show_item_opt:
4825             \skip_horizontal:n { -\l__enumext_fake_item_indent_viii_dim - \l__enumext_labelsep_v
4826           }
4827           {
4828             \tl_use:N \l__enumext_fake_item_indent_viii_tl
4829           }
4830 %%          First atempt here, need stop tag and set counter for enumi, and more...need test
4831 %%          \RenewDocumentCommand \item { o }
4832 %%            {
4833 %%              \tl_if_novalue:nTF {#1}
4834 %%                {
4835 %%                  \__enumext_item_std:w
4836 %%                }
4837 %%                {
4838 %%                  \__enumext_item_std:w [#1]
```

```
4839  %%                          }
4840  %%                  }
4841      }
```

(*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.

```
4842  \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
4843    {
4844        \__enumext_endminipage:
4845      \hbox_set_end:
4846      \int_set:Nn \hbadness { 10000 }
4847      \box_use_drop:N \l__enumext_item_text_viii_box
4848      \bool_if:NF \l__enumext_footnotes_key_bool
4849        {
4850          \__enumext_print_footnote:
4851        }
4852      \int_compare:nNnTF
4853        { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
4854        {
4855          \par\noindent
4856          \int_zero:N \l__enumext_item_column_pos_viii_int
4857        }
4858        {
4859          \skip_horizontal:N \l__enumext_columns_sep_viii_dim
4860        }
4861    }
```

(*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.

```
4862  \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
4863    {
4864      \int_compare:nNnT
4865        {
4866          \int_mod:nn
4867            { \g__enumext_item_count_all_viii_int }
4868            { \l__enumext_columns_viii_int }
4869        }
4870        =
4871        { 0 }
4872        {
4873          \para_end:
4874          \skip_vertical:n { -\l__enumext_itemsep_viii_skip }
4875          \skip_vertical:N \c_zero_skip
4876          \int_gzero:N \g__enumext_item_count_all_viii_int
4877        }
4878    }
```

(*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.

```
4879  \NewDocumentCommand \getkeyans { m }
4880    {
4881      \exp_args:Ne \__enumext_getkeyans_aux:n
4882        { \tl_to_str:e { \text_expand:n {#1} } }
4883    }
```

(*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.

```
4884  \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
4885    {
```

```
4886      \str_if_in:nnTF {#1} { : }
4887        {
4888          \use:e
4889            {
4890              \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
4891                { {##1} {##2} }
4892            }
4893          \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
4894        }
4895        { \msg_error:nnn { enumext } { missing-colon } {#1} }
4896      }
```

(*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*⟩.

```
4897  \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
4898    {
4899      \prop_if_exist:cTF { g__enumext_#1_prop }
4900        {
4901          \prop_item:cn { g__enumext_#1_prop }{#2}
4902        }
4903        {
4904          \msg_error:nnn { enumext } { undefined-storage-anskey } {#1}
4905        }
4906    }
```

(*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*⟩].

```
4907  \keys_define:nn  { enumext / print }
4908    {
4909      print*  .code:n     = \keys_precompile:neN { enumext / enumext* }
4910                             { \__enumext_filter_save_key:n {##1} }
4911                             \l__enumext_print_keyans_starred_tl, % starred cmd
4912      print*  .initial:n  = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4913      print-1 .code:n     = \keys_precompile:neN { enumext / level-1 }
4914                             { \__enumext_filter_save_key:n {##1} }
4915                             \l__enumext_print_keyans_i_tl,
4916      print-1 .initial:n  = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4917      print-2 .code:n     = \keys_precompile:neN { enumext / level-2 }
4918                             { \__enumext_filter_save_key:n {##1} }
4919                             \l__enumext_print_keyans_ii_tl,
4920      print-2 .initial:n  = { nosep, label=(\alph*), first=\small, font=\small },
4921      print-3 .code:n     = \keys_precompile:neN { enumext / level-3 }
4922                             { \__enumext_filter_save_key:n {##1} }
4923                             \l__enumext_print_keyans_iii_tl,
4924      print-3 .initial:n  = { nosep, label=\roman*., first=\small, font=\small },
4925      print-4 .code:n     = \keys_precompile:neN { enumext / level-4 }
4926                             { \__enumext_filter_save_key:n {##1} }
4927                             \l__enumext_print_keyans_iv_tl,
4928      print-4 .initial:n  = { nosep, label=\Alph*., first=\small, font=\small },
4929      print-* .code:n     = \keys_precompile:neN { enumext / enumext* }
4930                             { \__enumext_filter_save_key:n {##1} }
4931                             \l__enumext_print_keyans_vii_tl, % starred nested
4932      print-* .initial:n  = { nosep, label=\arabic*., first=\small, font=\small },
4933    }
```

🎯 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.

```
4934 \NewDocumentCommand \printkeyans { s O{} m }
4935   {
4936     \group_begin:
4937       \tl_use:N \l__enumext_print_keyans_i_tl
4938       \tl_use:N \l__enumext_print_keyans_ii_tl
4939       \tl_use:N \l__enumext_print_keyans_iii_tl
4940       \tl_use:N \l__enumext_print_keyans_iv_tl
4941       \tl_use:N \l__enumext_print_keyans_vii_tl
4942       \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
4943     \group_end:
4944   }
```

(*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.

```
4945 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
4946   {
4947     \seq_if_exist:cTF { g__enumext_#3_seq }
4948       {
4949         \seq_if_empty:cF { g__enumext_#3_seq }
4950           {
4951             %%\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*⟩.

```
4952             \bool_if:nTF {#1}
4953               {
4954                 \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
4955                   {
4956                     \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
4957                   }
4958                   {
4959                     \tl_use:N \l__enumext_print_keyans_starred_tl
4960                     \begin{enumext*}[#2]
4961                       \keys_set:nn { enumext / level-1 }{ base-fix }
4962                       \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4963                     \end{enumext*}
4964                   }
4965               }
```

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*⟩.

```
4966               {
4967                 \begin{enumext}[#2]
4968                   \keys_set:nn { enumext / enumext* }{ base-fix }
4969                   \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4970                 \end{enumext}
4971               }
4972           }
4973       }
4974       {
4975         \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
4976       }
4977   }
```

(*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 The function __enumext_filter_first_level:n will be in charge of filtering the ⟨*keys*⟩ passed to the
__enumext_filter_first_level_key:n environment enumext* and *"first level"* of the environment enumext.
__enumext_filter_first_level_pair:nn

```
4978 \cs_new:Npn \__enumext_filter_first_level:n #1
```

```
4979    {
4980      \use:e
4981        {
4982          \keyval_parse:NNn
4983            \__enumext_filter_first_level_key:n
4984            \__enumext_filter_first_level_pair:nn {#1}
4985        }
4986    }
```

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*.

```
4987  \cs_new:Npn \__enumext_filter_first_level_key:n #1
4988    {
4989      \str_case:nnF {#1}
4990        {
4991          { resume    } {}
4992          { resume*   } {}
4993        }
4994        { , { \exp_not:n {#1} } }
4995    }
```

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.

```
4996  \cs_new:Npn \__enumext_filter_first_level_pair:nn #1#2
4997    {
4998      \str_case:nnF {#1}
4999        {
5000          { series } {}
5001          { resume } {}
5002          { save-ans } {}
5003        }
5004        { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
5005    }
```

(*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.

```
5006  \keys_define:nn { enumext / meta-families }
5007    {
5008      enumext-1 .code:n =
5009              {
5010                \keys_set:ne { enumext / level-1 }
5011                  {
5012                    \__enumext_filter_first_level:n {#1}
5013                  }
5014              } ,
5015      enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
5016      enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
5017      enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
5018      keyans    .code:n = { \keys_set:nn { enumext / keyans  } {#1} } ,
5019      enumext*  .code:n =
5020              {
5021                \keys_set:ne { enumext / enumext* }
5022                  {
5023                    \__enumext_filter_first_level:n {#1}
5024                  }
5025              } ,
5026      keyans*   .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
5027      print*    .code:n = { \keys_set:nn { enumext / print   } { print*  = {#1} } } ,
5028      print-1   .code:n = { \keys_set:nn { enumext / print   } { print-1 = {#1} } } ,
5029      print-2   .code:n = { \keys_set:nn { enumext / print   } { print-2 = {#1} } } ,
5030      print-3   .code:n = { \keys_set:nn { enumext / print   } { print-3 = {#1} } } ,
5031      print-4   .code:n = { \keys_set:nn { enumext / print   } { print-4 = {#1} } } ,
5032      print-*   .code:n = { \keys_set:nn { enumext / print   } { print-* = {#1} } } ,
5033      unknown   .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
5034    }
```

We store them in the constant sequence \c__enumext_all_families_seq separated by commas.

```
5035  \seq_const_from_clist:Nn \c__enumext_all_families_seq
5036    {
5037      enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
5038      keyans*, print-1, print-2, print-3, print-4, print-*, print*,
5039    }
```

\setenumext

Now we define the user command \setenumext.

```
5040  \NewDocumentCommand \setenumext { O{enumext,1} +m }
5041    {
5042      \seq_clear:N \l__enumext_setkey_tmpa_seq
5043      \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
5044      \int_set:Nn \l__enumext_setkey_tmpa_int
5045        {
5046          \seq_count:N \l__enumext_setkey_tmpb_seq
5047        }
5048      \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
5049        {
5050          \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
5051          \seq_map_function:NN \l__enumext_setkey_tmpb_seq \__enumext_set_parse:n
5052          \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
5053            {
5054              \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
5055            }
5056        }
5057        {
5058          \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
5059        }
5060      \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
5061        { \seq_map_inline:Nn \c__enumext_all_families_seq }
5062        { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
5063        {
5064          \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
5065        }
5066    }
```

(*End of definition for* \setenumext. *This function is documented on page* *6*.)

__enumext_set_parse:n
__enumext_set_error:nn

Internal functions used by the \setenumext command.

```
5067  \cs_new_protected:Npn \__enumext_set_parse:n #1
5068    {
5069      \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
5070      \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
5071        { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
5072      \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
5073        {
5074          \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
5075            { \tl_trim_spaces:n {#1} }
5076        }
5077        { \__enumext_set_error:nn {#1} { } }
5078    }
5079  \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
5080    { \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.

```
5081  \prop_const_from_keyval:Nn \c__enumext_meta_paths_prop
5082    {
5083      {enumext,1} = level-1,
5084      {enumext,2} = level-2,
5085      {enumext,3} = level-3,
5086      {enumext,4} = level-4,
5087      {enumext*}  = enumext*
5088    }
```

Now we create the user command taking care that unknown cannot be passed as an argument.

```
5089  \NewDocumentCommand \setenumextmeta { s O{enumext,1} m +m }
5090    {
5091      \str_if_eq:eeTF { \tl_trim_spaces:n {#3} } { unknown }
5092        { \msg_error:nn { enumext } { prohibited-unknown } }
5093        {
5094          \bool_if:nTF {#1}
```

```
5095          {
5096            \int_step_inline:nn { 4 }
5097              { \__enumext_add_meta_key:nnn { enumext, ##1 } {#3} {#4} }
5098            \__enumext_add_meta_key:nnn { enumext* } {#3} {#4}
5099          }
5100          { \__enumext_add_meta_key:nnn {#2} {#3} {#4} }
5101      }
5102   }
```

The internal functions __enumext_add_meta_key:nnn and __enumext_def_meta_key:nnn will check the optional argument and create the *"meta-key"*.

```
5103 \cs_new_protected:Npn \__enumext_add_meta_key:nnn #1
5104   {
5105     \tl_set:Nn \l__enumext_meta_path_tl {#1}
5106     \tl_replace_all:Nnn \l__enumext_meta_path_tl { ~ } {}
5107     \prop_get:NVNTF
5108       \c__enumext_meta_paths_prop \l__enumext_meta_path_tl \l__enumext_meta_path_tl
5109       { \__enumext_def_meta_key:Vnn \l__enumext_meta_path_tl }
5110       {
5111         \msg_error:nnn { enumext } { unknown-set } {#1}
5112         \use_none:nn
5113       }
5114   }
5115 \cs_new_protected:Npn \__enumext_def_meta_key:nnn #1#2#3
5116   {
5117     \bool_lazy_or:nnTF
5118       { \keys_if_exist_p:nn { enumext / #1 } {#2} }
5119       { \keys_if_exist_p:nn { enumext / enumext* } {#2} }
5120       { \msg_error:nnn { enumext } { already-defined } {#2} }
5121       {
5122         \keys_define:nn { enumext / #1 }
5123           {
5124             #2 .meta:n = {#3},
5125             #2 .value_forbidden:n = true
5126           }
5127       }
5128   }
5129 \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.

\foreachkeyans
__enumext_parse_foreach_keys:nn
__enumext_parse_foreach_keys:n
__enumext_foreach_keyans:nn
__enumext_foreach_add_body:n

We define a set of ⟨*keys*⟩ for command and we will save the default values of these in \g__enumext_-foreach_default_keys_tl to avoid the use of group.

```
5130 \keys_define:nn { enumext / foreach }
5131   {
5132     before  .tl_set:N  = \l__enumext_foreach_before_tl,
5133     before  .value_required:n = true,
5134     after   .tl_set:N  = \l__enumext_foreach_after_tl,
5135     after   .value_required:n = true,
5136     start   .int_set:N = \l__enumext_foreach_start_int,
5137     start   .value_required:n = true,
5138     stop    .int_set:N = \l__enumext_foreach_stop_int,
5139     stop    .value_required:n = true,
5140     step    .int_set:N = \l__enumext_foreach_step_int,
5141     step    .value_required:n = true,
5142     wrapper .cs_set_protected:Np = \__enumext_foreach_wrapper:n #1,
5143     wrapper .value_required:n = true,
5144     sep     .tl_set:N  = \l__enumext_foreach_sep_tl,
5145     sep     .value_required:n = true,
5146     unknown .code:n    = { \__enumext_parse_foreach_keys:n {#1} }
5147   }
5148 \keys_precompile:nnN { enumext / foreach }
5149   {
5150     before={},after={},start=1,step=1,stop=0,wrapper=#1,sep=
5151   }
5152   \g__enumext_foreach_default_keys_tl
```

Functions for handling unknown ⟨*keys*⟩.

```
5153 \cs_new_protected:Npn \__enumext_parse_foreach_keys:nn #1#2
5154   {
5155     \tl_if_blank:nTF {#2}
5156       {
5157         \msg_error:nnn { enumext } { for-key-unknown } {#1}
5158       }
5159       {
5160         \msg_error:nnnn { enumext } { for-key-value-unknown } {#1} {#2}
5161       }
5162   }
5163 \cs_new_protected:Npn \__enumext_parse_foreach_keys:n #1
5164   {
5165     \exp_args:NV \__enumext_parse_foreach_keys:nn \l_keys_key_str {#1}
5166   }
```

We create the command.

```
5167 \NewDocumentCommand \foreachkeyans { +O{} m }
5168   {
5169     \__enumext_foreach_keyans:nn {#1} {#2}
5170   }
```

Finally the internal functions __enumext_foreach_keyans:nn and __enumext_foreach_add_body:n
will loop through the prop list and print the contents.

```
5171 \cs_new_protected:Npn \__enumext_foreach_keyans:nn #1 #2
5172   {
5173     \tl_use:N \g__enumext_foreach_default_keys_tl
5174     \keys_set:nn { enumext / foreach } {#1}
5175     \tl_set:Nn \l__enumext_foreach_name_prop_tl {#2}
5176     \prop_if_exist:cF { g__enumext_#2_prop }
5177       {
5178         \msg_error:nnn { enumext } { undefined-storage-anskey } {#2}
5179       }
5180     \int_compare:nNnT { \l__enumext_foreach_stop_int } = { 0 }
5181       {
5182         \int_set:Nn \l__enumext_foreach_stop_int
5183           { \prop_count:c { g__enumext_#2_prop } }
5184       }
5185     \seq_clear:N \l__enumext_foreach_print_seq
5186     \int_step_function:nnnN
5187       { \l__enumext_foreach_start_int }
5188       { \l__enumext_foreach_step_int }
5189       { \l__enumext_foreach_stop_int }
5190       \__enumext_foreach_add_body:n
5191     \seq_use:NV \l__enumext_foreach_print_seq \l__enumext_foreach_sep_tl
5192   }
5193 \cs_new_protected:Npn \__enumext_foreach_add_body:n #1
5194   {
5195     \seq_put_right:Ne \l__enumext_foreach_print_seq
5196       {
5197         \exp_not:V \l__enumext_foreach_before_tl
5198         \__enumext_foreach_wrapper:n
5199           {
5200             \prop_item:cn { g__enumext_ \l__enumext_foreach_name_prop_tl _prop }{#1}
5201           }
5202         \exp_not:V \l__enumext_foreach_after_tl
5203       }
5204   }
```

(*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.

```
5205 \msg_new:nnn { enumext } { package-load }
5206   {
5207     The ~ '#1' ~ package ~ is ~ already ~ loaded.
5208   }
5209 \msg_new:nnn { enumext } { package-not-load }
5210   {
5211     The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
5212   }
```

```
5213 \msg_new:nnn { enumext } { package-load-foot }
5214   {
5215     The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
5216   }
```

Message used in the creation of counters by enumext package.

```
5217 \msg_new:nnn { enumext } { counters }
5218   {
5219     The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
5220     package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
5221   }
```

Message used by align and mark-pos keys.

```
5222 \msg_new:nnn { enumext } { unknown-choice }
5223   {
5224     The ~ value ~ '#3' ~ for ~ '#1' ~ key ~ is ~ invalid ~ use ~ ('#2').
5225   }
```

Message used by reserved anskey* environment by enumext package.

```
5226 \msg_new:nnnn { enumext } { anskey-env-error }
5227   {
5228     The ~ '#1' ~ environment ~is ~ reserved ~ by ~\\
5229     'enumext' ~ package, ~ It~ is~ already~ defined.
5230   }
5231   {
5232     The ~ anskey* ~ environment ~ is ~ defined ~ internally ~
5233     for ~ the ~ 'save-ans' ~ key.\\
5234   }
```

Message used in the creation of ⟨*prop list*⟩ by enumext package.

```
5235 \msg_new:nnn { enumext } { store-prop }
5236   {
5237     * ~ Package ~ enumext: ~ Creating ~
5238     \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
5239   }
5240 \msg_new:nnn { enumext } { store-seq }
5241   {
5242     * ~ Package ~ enumext: ~ Creating ~
5243     \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
5244   }
5245 \msg_new:nnn { enumext } { store-int }
5246   {
5247     * ~ Package ~ enumext: ~ Creating ~
5248     \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_context:.
5249   }
5250 \msg_new:nnn { enumext } { prop-seq-int-hook }
5251   {
5252     * ~ Package ~ enumext: ~ Elements ~ in ~
5253     \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
5254     * ~ Package ~ enumext: ~ Elements ~ in ~
5255     \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
5256     * ~ Package ~ enumext: ~ Value ~ off ~
5257     \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
5258   }
5259 \msg_new:nnn { enumext } { item-answer-hook }
5260   {
5261     * ~ Package ~ enumext: ~ Value ~ off ~
5262     \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\\
5263     * ~ Package ~ enumext: ~ Value ~ off ~
5264     \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\\
5265     * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
5266   }
```

Message used by [⟨*key = val*⟩] system and \setenumext command.

```
5267 \msg_new:nnn { enumext } { invalid-key }
5268   {
5269     The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
5270   }
5271 \msg_new:nnn { enumext } { unknown-key-family }
5272   {
5273     Unknown~key~family~`\l_keys_key_str'~for~enumext.
5274   }
```

Messages used in length calculation.

```
5275  \msg_new:nnn { enumext } { width-negative }
5276    {
5277      Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\\
5278      The ~ key ~ '#1'~ accepts ~ values  ~ >= ~ 0pt.
5279    }
5280  \msg_new:nnn { enumext } { width-zero }
5281    {
5282      Invalid ~ '#1=#2' ~ \msg_line_context:.\\
5283      The ~ key ~ '#1'~ accepts ~ values  ~ > ~ 0pt.
5284    }
```

Messages used by show-length key in enumext.

```
5285  \msg_new:nnn { enumext } { list-lengths }
5286    {
5287      **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
5288      \__enumext_show_length:nnn { dim  } { labelsep      } {#1}
5289      \__enumext_show_length:nnn { dim  } { labelwidth    } {#1}
5290      \__enumext_show_length:nnn { dim  } { itemindent    } {#1}
5291      \__enumext_show_length:nnn { dim  } { leftmargin    } {#1}
5292      \__enumext_show_length:nnn { dim  } { rightmargin   } {#1}
5293      \__enumext_show_length:nnn { dim  } { listparindent } {#1}
5294      \__enumext_show_length:nnn { skip } { topsep     } {#1}
5295      \__enumext_show_length:nnn { skip } { parsep     } {#1}
5296      \__enumext_show_length:nnn { skip } { partopsep } {#1}
5297      \__enumext_show_length:nnn { skip } { itemsep    } {#1}
5298      ****************************************************
5299    }
```

Messages used by show-length key in enumext*, keyans* and keyans.

```
5300  \msg_new:nnn { enumext } { list-lengths-not-nested }
5301    {
5302      **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
5303      \__enumext_show_length:nnn { dim  } { labelsep      } {#1}
5304      \__enumext_show_length:nnn { dim  } { labelwidth    } {#1}
5305      \__enumext_show_length:nnn { dim  } { itemindent    } {#1}
5306      \__enumext_show_length:nnn { dim  } { leftmargin    } {#1}
5307      \__enumext_show_length:nnn { dim  } { rightmargin   } {#1}
5308      \__enumext_show_length:nnn { dim  } { listparindent } {#1}
5309      \__enumext_show_length:nnn { skip } { topsep     } {#1}
5310      \__enumext_show_length:nnn { skip } { parsep     } {#1}
5311      \__enumext_show_length:nnn { skip } { partopsep } {#1}
5312      \__enumext_show_length:nnn { skip } { itemsep    } {#1}
5313      ****************************************************
5314    }
```

Messages used by ref key.

```
5315  \msg_new:nnn { enumext } { key-ref-empty }
5316    {
5317      Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
5318    }
```

Messages used by save-ans key.

```
5319  \msg_new:nnn { enumext } { save-ans-empty }
5320    {
5321      Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
5322    }
5323  \msg_new:nnn { enumext } { save-ans-log }
5324    {
5325      * ~ Package ~ enumext: ~ Start ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5326    }
5327  \msg_new:nnn { enumext } { save-ans-log-hook }
5328    {
5329      * ~ Package ~ enumext: ~ Stop ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5330    }
5331  \msg_new:nnn { enumext } { save-ans-hook }
5332    {
5333      Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
5334    }
```

Messages used by the internal system to check answer used by check-ans key.

```
5335  \msg_new:nnn { enumext } { need-save-ans }
5336    {
```

```
5337        Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
5338      }
5339  \msg_new:nnn { enumext } { items-same-answer }
5340      {
5341        ***************************************\\
5342        * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~
5343        for ~ \c_left_brace_str #2 \c_right_brace_str\\
5344        * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~
5345        'OK', ~ all ~ items ~ with ~ answer.\\
5346        ***************************************
5347      }
5348  \msg_new:nnn { enumext } { item-greater-answer }
5349      {
5350        Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
5351        started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
5352        Items ~ > ~ Answers.
5353      }
5354  \msg_new:nnn { enumext } { item-less-answer }
5355      {
5356        Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
5357        started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
5358        Items ~ < ~ Answers.
5359      }
```

Messages used by the internal system to check for *"starred"* `\item*` and `\anspic*` commands.

```
5360  \msg_new:nnn { enumext } { missing-starred }
5361      {
5362        Missing ~ '\c_backslash_str #1*' ~ #2.
5363      }
5364  \msg_new:nnn { enumext } { many-starred }
5365      {
5366        Many ~ '\c_backslash_str #1*' ~ #2.
5367      }
```

Messages used by `\printkeyans*` command.

```
5368  \msg_new:nnn { enumext } { print-starred }
5369      {
5370        \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
5371        #2 ~ environment ~  \msg_line_context:.
5372      }
```

Message for the nesting depth of the environment `enumext`.

```
5373  \msg_new:nnn { enumext } { list-too-deep }
5374      {
5375        Too ~ deep ~ nesting  ~ for ~ 'enumext' ~ \msg_line_context:.~ \\
5376        The ~ maximum  ~ level  ~ of  ~ nesting  ~ is ~ 4.
5377      }
```

Messages used by `\anskey`, `anskey*` and `\anspic` commands.

```
5378  \msg_new:nnn { enumext } { anskey-unnumber-item }
5379      {
5380        Can't ~ store ~ with ~ a ~ unnumbered ~ \c_backslash_str item ~ \msg_line_context:.
5381      }
5382  \msg_new:nnn { enumext } { anskey-already-stored }
5383      {
5384        Content ~ already ~ stored ~ for ~ this ~ \c_backslash_str item ~ \msg_line_context:.
5385      }
5386  \msg_new:nnn { enumext } { anskey-empty-arg }
5387      {
5388        Can't ~ store ~ empty ~ content ~ \msg_line_context:.
5389      }
5390  \msg_new:nnn { enumext } { anskey-wrong-place }
5391      {
5392        Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
5393        '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5394      }
5395  \msg_new:nnn { enumext } { anskey-nested }
5396      {
5397        The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
5398      }
5399  \msg_new:nnn { enumext } { anskey-math-mode }
5400      {
5401        #1 ~ can't ~ work ~ in ~ math ~ mode ~ \msg_line_context:.
```

```
5402        }
5403   \msg_new:nnn { enumext } { anskey-env-wrong }
5404     {
5405        The ~ environment ~ anskey* ~ cannot ~ use ~ in ~ '#1' ~ \msg_line_context:.
5406     }
5407   \msg_new:nnn { enumext } { anspic-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 } { command-wrong-place }
5413     {
5414        Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
5415        '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
5416     }
5417   \msg_new:nnnn { enumext } { anskey-env-key-unknown }
5418     {
5419        The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5420        'anskey*' ~ and ~ is ~ being ~ ignored.
5421     }
5422     {
5423        The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5424        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5425     }
5426   \msg_new:nnnn { enumext } { anskey-env-key-value-unknown }
5427     {
5428        The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5429        'anskey*' ~ and ~ is ~ being ~ ignored.
5430     }
5431     {
5432        The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5433        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5434     }
5435   \msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
5436     { The ~ key ~'#1'~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.}
5437     {
5438        The ~ command ~'\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5439        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5440     }
5441   \msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
5442     { The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignor
5443     {
5444        The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5445        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5446     }
```

Messages used by keyans, keyans* and keyanspic environment.

```
5447   \msg_new:nnn { enumext } { keyans-nested }
5448     {
5449        The ~ environment ~ 'keyans' ~ can't ~ be  ~ nested  ~ \msg_line_context:.
5450     }
5451   \msg_new:nnn { enumext } { keyans-wrong-level }
5452     {
5453        Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:.~ \\
5454        The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5455     }
5456   \msg_new:nnn { enumext } { wrong-place }
5457     {
5458        Wrong ~ place ~ for ~ '#1' ~ environment ~\msg_line_context:.~ \\
5459        '#1' ~ is ~ only ~ found ~ with ~ '#2' ~  in  ~  'enumext.
5460     }
5461   \msg_new:nnn { enumext } { keyanspic-nested }
5462     {
5463        The ~ environment ~ 'keyanspic' ~ can't ~ be  ~ nested~ \msg_line_context:.~.
5464     }
5465   \msg_new:nnn { enumext } { keyanspic-wrong-level }
5466     {
5467        Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:.~ \\
5468        The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5469     }
5470   \msg_new:nnn { enumext } { keyanspic-item-cmd }
5471     {
```

```
5472        Can't ~ use  ~ \c_backslash_str item ~ in ~ keyanspic ~ \msg_line_context:.
5473      }
5474  \msg_new:nnnn { enumext } { keyans-unknown-key }
5475      {
5476        The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5477        '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5478      }
5479      {
5480        The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5481        ~ have ~ a ~ key ~ called ~'#1'.\\
5482        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5483      }
5484  \msg_new:nnnn { enumext } { keyans-unknown-key-value }
5485      {
5486        The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5487        '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5488      }
5489      {
5490        The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5491        ~ have ~ a ~ key ~ called ~'#1'.\\
5492        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5493      }
```

Message used by unknown ⟨*keys*⟩ in enumext*. environment.

```
5494  \msg_new:nnnn { enumext } { starred-unknown-key }
5495      {
5496        The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5497        '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5498      }
5499      {
5500        The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5501        ~ have ~ a ~ key ~ called ~'#1'.\\
5502        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5503      }
5504  \msg_new:nnnn { enumext } { starred-unknown-key-value }
5505      {
5506        The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5507        '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5508      }
5509      {
5510        The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5511        ~ have ~ a ~ key ~ called ~'#1'.\\
5512        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5513      }
```

Message used by unknown ⟨*keys*⟩ in enumext environment.

```
5514  \msg_new:nnnn { enumext } { standar-unknown-key }
5515      {
5516        The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_tl
5517        ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5518      }
5519      {
5520        The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5521        ~ have ~ a ~ key ~ called ~'#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5522        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5523      }
5524  \msg_new:nnnn { enumext } { standar-unknown-key-value }
5525      {
5526        The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_t
5527        ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5528      }
5529      {
5530        The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5531        ~ have ~ a ~ key ~ called ~'#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5532        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5533      }
```

Message used by unknown ⟨*keys*⟩ in \foreachkeyans.

```
5534  \msg_new:nnnn { enumext } { for-key-unknown }
5535      { The~key~'#1'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored.}
5536      {
5537        The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
```

```
5538        Check~that~you~have~spelled~the~key~name~correctly.
5539    }
5540  \msg_new:nnnn { enumext } { for-key-value-unknown }
5541    { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored. }
5542    {
5543      The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5544      Check~that~you~have~spelled~the~key~name~correctly.
5545    }
```

Messages used by \getkeyans command.

```
5546  \msg_new:nnn { enumext } { undefined-storage-anskey }
5547    {
5548      Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
5549    }
```

Messages used by \miniright command.

```
5550  \msg_new:nnn { enumext } { missing-miniright }
5551    {
5552      Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\\
5553      The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
5554    }
5555  \msg_new:nnn { enumext } { wrong-miniright-place }
5556    {
5557      Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
5558      Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
5559    }
5560  \msg_new:nnn { enumext } { wrong-miniright-use }
5561    {
5562      Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
5563      '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
5564    }
5565  \msg_new:nnn { enumext } { wrong-miniright-starred }
5566    {
5567      Can't ~ use  ~ \c_backslash_str miniright ~ in ~ starred ~ environments ~ \msg_line_context:.
5568    }
5569  \msg_new:nnn { enumext } { many-miniright-used }
5570    {
5571      Can't ~ use  ~ \c_backslash_str miniright ~ more ~ than ~ once ~  \msg_line_context:.
5572    }
```

Messages used by \setenumextmeta command.

```
5573  \msg_new:nnn { enumext } { unknown-set }
5574    {
5575      Argument ~ [#1] ~ is ~ unknown ~ by ~  \c_backslash_str setenumextmeta ~ \msg_line_context:.
5576    }
5577  \msg_new:nnn { enumext } { already-defined }
5578    {
5579      The ~ key ~ '#1' ~ is ~ already ~ defined ~ \msg_line_context:.
5580    }
5581  \msg_new:nnn { enumext } { prohibited-unknown }
5582    {
5583      The ~ name ~ 'unknown' ~ can't ~ be ~ chosen~ for ~ a ~ meta ~ key ~ \msg_line_context:.
5584    }
```

Messages used by enumext* and keyans* environments.

```
5585  \msg_new:nnn { enumext } { nested }
5586    {
5587      The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ \msg_line_con
5588    }
5589  \msg_new:nnn { enumext } { nested-horizontal }
5590    {
5591      The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ in ~ '#1' ~ '
5592    }
5593  \msg_new:nnn { enumext } { item-joined }
5594    {
5595      Items ~ joined ~ (#1) ~ > ~ #2  ~ columns ~\msg_line_context:.
5596    }
5597  \msg_new:nnn { enumext } { item-joined-columns }
5598    {
5599      Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.
5600    }
```

12.51 Finish package

Finish package implementation.

```
5601 \file_input_stop:
5602 ⟨/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.