enumext
ENUMERATE EXERCISE SHEETS

V1.0   2024-06-24[*]

©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)   C)

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

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

1.3.2 Public dimension

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

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

1.3.3 Support for multicol

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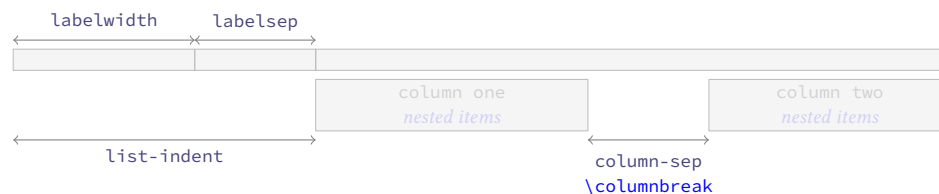

Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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

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

1.3.4 Support for `minipage`

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

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

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

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

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

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

1.3.6 Support for \footnote

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

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

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

2 The environments provided

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

enumext
enumext*

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

2.1 The environment enumext

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

Example with columns=2

1. This text is in the first level.

   (a) This text is in the second level.

      i. This text is in the third level.

A. This text is in the fourth level.

X This text is in the first level.

★ 2. This text is in the first level.

2.2 The environment enumext*

The enumext* is a *horizontal list environment* similar to the enumerate* environment provided by the enumitem package or task environment provided by the task package , \item and \item[⟨custom⟩] work as usual. The options can be configured globally using \setenumext command and locally using [⟨key = val⟩] in the environment.

Some considerations to take into account for this environment:

- The environment cannot be nested within itself or in the environment keyans*, but it can be nested within enumext and vice versa.

- Each *"item"* in the environment is placed within a minipage environment whose *width* is stored in the dimension \itemwidth that NOT includes labelwith, labelsep, only the *width of the content*.

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

Example with columns=2

1. This text is in the first level.

X This text is in the first level.

2. This text is in the first level.

★ 3. 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 two columns (187.14374pt)   8. The ninth

Z The tenth (84.76483pt)

3 The command `\setenumext`

`\setenumext`

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

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

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

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

4 The command `\setenumextmeta`

`\setenumextmeta`

`\setenumextmeta {`⟨*key name*⟩`}{`⟨*key-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*⟩} $\qquad$ 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*⟩} $\qquad$ default: *not used*

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

💣 The keys `mini-right` and `mini-right*` has a *slightly different* implementation, the argument {⟨*content*⟩} is saved in a box and then printed outside the environment using *hooks*.

6 The storage system

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

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

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

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

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

6.1 Keys for storage system

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

`save-ans` = {⟨*store name*⟩} $\qquad$ 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*⟩} $\qquad$ 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*⟩} $\qquad$ 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   * B) Correct choice

   C) Choice   D) Choice

   E) Choice

2. Text containing a question and image.

   A) Choice

   B) Choice

   C) Choice

   D) Choice

   * E) [note] Correct choice

Some text

6.5 The environment keyanspic

`keyanspic`

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

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

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

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

6.5.1 The command \anspic

\anspic

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

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

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

Example

```
\begin{enumext}[save-ans=test,show-ans,nosep]
  \item Question with images.
    \begin{keyanspic}[3,2]
      \anspic{\includegraphics[scale=0.15]{example-image-a}}
      \anspic{\includegraphics[scale=0.15]{example-image-b}}
      \anspic{\includegraphics[scale=0.15]{example-image-a}}
      \anspic{\includegraphics[scale=0.15]{example-image-a}}
      \anspic*[note]{\includegraphics[scale=0.15]{example-image-a}}
    \end{keyanspic}
\end{enumext}
```
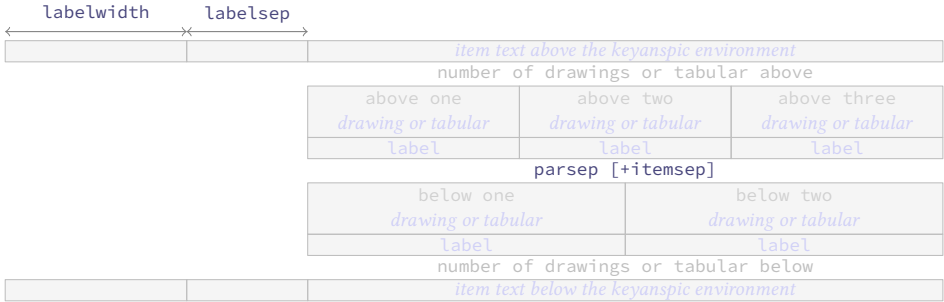
1. Question with images.
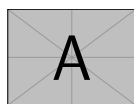
A)
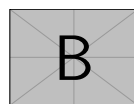
B)
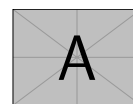
C)

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 \printkeyans

\printkeyans

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

The command `\printkeyans` prints *"all stored content"* in *sequence* {⟨*store name*⟩} defined by `save-ans` key placing this inside the `enumext` environment or the `enumext*` environment if the *starred argument* '`*`' is used. The *"stored content"* can only be accessed *after* it is stored in the *sequence*, if {⟨*store name*⟩} does not exist the command will return an error.

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

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

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

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

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

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

The default values for the *"first level"* of `\printkeyans` commands and `\printkeyans*` are established using `\setenumext[`⟨*print*, *1*⟩`]{`⟨*keys*⟩`}` and `\setenumext[`⟨*print**⟩`]{`⟨*keys*⟩`}`. If we need to set the ⟨*keys*⟩ for the environment `enumext` "saved" in the *sequence* `{`⟨*store name*⟩`}` we will use `\setenumext[`⟨*print*, *level*⟩`]{`⟨*keys*⟩`}` and if we need to set the ⟨*keys*⟩ for the environment `enumext*` "saved" in the *sequence* `{`⟨*store name*⟩`}` we will use `\setenumext[`⟨*print*, ***⟩`]{`⟨*keys*⟩`}`.

Example

```
\begin{enumext}[save-ans=sample,columns=2,show-pos=true,nosep,save-ref=true]
  \item Factor $3x+3y+3z$. \anskey{$3(x+y+z)$}
  \item True False

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

  \item Related to Linux

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

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

\printkeyans{sample}
```

1. Factor $3x + 3y + 3z$.
   [1] $3(x + y + z)$
2. True False
   (a) LaTeX2e is cool?
       [2] Very True!
3. Related to Linux
   (a) You use linux?

[3] Yes
(b) Rate the following package and class
    i. `xsim`
       [4] very good
    ii. `exsheets`
       [5] obsolete

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

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

7   Full examples

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

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

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

Example 1

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

---

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

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

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

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

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

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

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

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

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

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

Example 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. $\angle EDF = 45°$

(A) I only

(B) II only

(C) I and II only

(D) I and III only

(E) I, II, and III

3. Third type of questions

    (1) $2\alpha + 2\delta = 90°$

    (2) $\angle EDF = 45°$

(A) value

(B) value

(C) value

(D) value

(E) value

4. Question with image and label below:

(A)

(B)

(C)

(D)

(E)
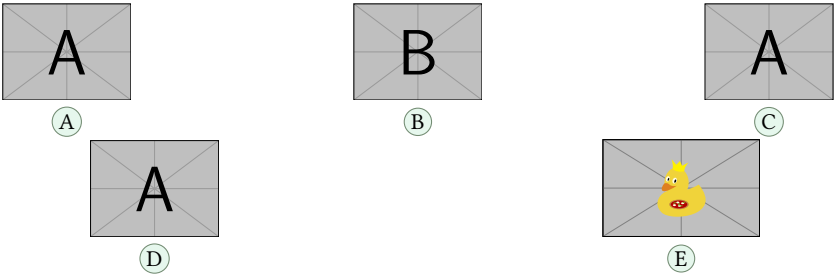
5. Question with image on left side:

(A) value

(B) value

(C) value

(D) correct

(E) value

Test keys

1. B, $x = 5$ *

2. D *

3. C, some note *

4. E, A duck *

5. D, other note *

Example 4

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

1. Factor $x^2 - 2x + 1$

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

The following questions need to be cuaqtified :)

3. True False

    (a) $\alpha > \delta$

    (b) LaTeX2e is cool?

4. Related to Linux

    (a) You use linux?

    (b) Usually uses the package manager?

    (c) Rate the following package and class

       i.   `xsim-exam`

       ii. `xsim`

       iii. `exsheets`

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

1. $(x - 1)^2$ *

2. $3(x + y + z)$ *

3. (a) False *

   (b) Very True! *

4. (a) Yes *

   (b) Yes, `dnf` *

   (c) i. doesn't exist for now :(*

      ii. very good *

      iii. obsolete *

Example 5

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

1

Which choice best describes what happens in the passage?

A) One character argues with another character who intrudes on her home.

B) One character receives a surprising request from another character.

C) One character reminisces about choices she has made over the years.

D) One character criticizes another character for pursuing an unexpected course of action.

2

Which choice best describes what happens in the passage?

A) One character argues with another charac-

ter 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 2_ε: An Unofficial Reference Manual". Available from ctan, `https://ctan.org/pkg/latex2e-help-texinfo`, 2024.

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

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

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

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

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

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

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

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

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

10 Change history

v1.0 2024-06-24 – 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    {2024-06-24}
7    {1.0}
8    {Enumerate exercise sheets}
```

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

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

12.4   Definition of variables

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

\l__enumext_level_int

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
```

\l__enumext_level_h_int
\l__enumext_anskey_level_int
\l__enumext_keyans_level_int
\l__enumext_keyans_level_h_int
\l__enumext_keyans_pic_level_int

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

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

Internal variables used by functions __enumext_is_not_nested:, __enumext_is_on_first_-level: and __enumext_keyans_name_and_start: (§12.5.1).

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

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

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

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

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

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

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

\c__enumext_counter_style_tl
\l__enumext_ref_key_arg_tl
\l__enumext_ref_the_count_tl
\l__enumext_the_counter_X_tl
\l__enumext_renew_the_count_X_tl

Internal variables used by ref key (§12.13).

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

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

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

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

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

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

\l__enumext_current_widest_dim
\g__enumext_counter_styles_tl
\g__enumext_widest_label_tl
\l__enumext_label_width_by_box

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

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

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

\l__enumext_leftmargin_tmp_X_bool
\l__enumext_leftmargin_tmp_X_dim
\l__enumext_leftmargin_X_dim
\l__enumext_itemindent_X_dim

The boolean variable \l__enumext_leftmargin_tmp_X_bool and the dimensional variable \l__enumext_leftmargin_tmp_X_dim are used by the list-indent key (§12.17). The variables \l__enumext_leftmargin_X_dim and \l__enumext_itemindent_X_dim are used and set by the function __enumext_calc_hspace:NNNNNNNNNNNNN (§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

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   }
84 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }
```

(*End of definition for* \l__enumext_multicols_above_X_skip *and* \l__enumext_multicols_below_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).

```
85 \int_new:N  \g__enumext_minipage_stat_int
86 \skip_new:N \l__enumext_minipage_left_skip
87 \skip_new:N \l__enumext_minipage_right_skip
88 \skip_new:N \l__enumext_minipage_after_skip
89 \skip_new:N \g__enumext_minipage_right_skip
90 \skip_new:N \g__enumext_minipage_after_skip
91 \cs_set_protected:Npn \__enumext_tmp:n #1
92   {
93     \dim_new:c  { l__enumext_minipage_left_#1_dim    }
94     \bool_new:c { l__enumext_minipage_active_#1_bool }
95   }
96 \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).

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

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

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

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

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

The variables `\l__enumext_store_current_label_tl` and `\l__enumext_store_current_opt_-arg_tl` save the *current label* and *optional argument* of `\item*` (§12.36) and `\anspic*` (§12.40.1) 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.

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

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

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

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

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

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

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

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

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

```
127  \tl_new:N  \l__enumext_print_keyans_starred_tl
128  \str_new:N \l__enumext_mark_position_str
129  \tl_new:N  \g__enumext_item_symbol_aux_tl
130  \cs_set_protected:Npn \__enumext_tmp:n #1
131    {
132      \tl_new:c   { l__enumext_print_keyans_#1_tl          }
133      \tl_new:c   { l__enumext_store_save_key_#1_tl         }
134      \bool_new:c { l__enumext_store_save_key_#1_bool      }
135      \bool_new:c { l__enumext_store_upper_level_#1_bool }
136    }
137  \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {#1} }
```

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

Internal variables used by `keyanspic` environment (§12.40.2).

```
138  \seq_new:N  \l__enumext_keyans_pic_body_seq
139  \dim_new:N  \l__enumext_keyans_pic_width_dim
140  \int_new:N  \l__enumext_keyans_pic_above_int
141  \int_new:N  \l__enumext_keyans_pic_below_int
142  \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.

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

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

The boolean variable `\l__enumext_hyperref_bool` will determine if the `hyperref` package is present or load in memory (§12.8). The boolean variable `\l__enumext_footnotes_key_bool` determine if `hyperref` is load with key `hyperfootnotes=true`.

```
151  \bool_new:N \l__enumext_hyperref_bool
152  \bool_new:N \l__enumext_footnotes_key_bool
```

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

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

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

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

Internal variables used for redefinition of `\footnote` (§12.42).

```
161  \int_new:N \g__enumext_footnote_int
162  \seq_new:N \g__enumext_footnote_arg_seq
163  \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`.)

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

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

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

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

12.5 Some utility functions

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

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

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

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

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

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

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

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

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

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

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

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

(*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.

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

(*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.

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

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

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.

```
228  \cs_new_protected:Nn \__enumext_is_not_nested:
229    {
230      \str_case:en { \@currenvir }
231        {
232          {enumext}
233            {
234              \tl_set:Nn \l__enumext_envir_name_tl { enumext }
235              \bool_lazy_and:nnT
236                { \bool_not_p:n { \g__enumext_standar_bool } }
237                { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
238                {
239                  \bool_gset_true:N \g__enumext_standar_bool
240                }
241            }
242          {enumext*}
243            {
244              \tl_set:Nn \l__enumext_envir_name_tl { enumext* }
245              \bool_lazy_and:nnT
246                { \bool_not_p:n { \g__enumext_starred_bool } }
247                { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
248                {
249                  \bool_gset_true:N \g__enumext_starred_bool
250                }
251            }
252        }
253    }
```

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.

```
254  \cs_new_protected:Nn \__enumext_is_on_first_level:
255    {
256      \bool_lazy_all:nT
257        {
258          { \bool_if_p:N \g__enumext_standar_bool }
259          { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
260          { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
261        }
262        {
263          \bool_set_true:N \l__enumext_standar_first_bool
264          \bool_set_true:N \l__enumext_anskey_env_bool
265          \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
266          \tl_gset:Ne \g__enumext_start_line_tl
267            {
268              on ~ line ~ \exp_not:V \inputlineno
269            }
270        }
271      \bool_lazy_all:nT
272        {
273          { \bool_if_p:N \g__enumext_starred_bool }
274          { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
```

```
275              { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
276          }
277          {
278              \bool_set_true:N \l__enumext_starred_first_bool
279              \bool_set_true:N \l__enumext_anskey_env_bool
280              \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
281              \tl_gset:Ne \g__enumext_start_line_tl
282                  {
283                      on ~ line ~ \exp_not:V \inputlineno
284                  }
285          }
286      }
```

(*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.

```
287 \cs_new_protected:Nn \__enumext_keyans_name_and_start:
288     {
289         \str_case:en { \@currenvir }
290             {
291                 {keyans}
292                     {
293                         \tl_set:Nn \l__enumext_envir_name_tl { keyans }
294                         \tl_set:Ne \l__enumext_check_start_line_env_tl
295                             {
296                                 in ~ 'keyans' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
297                             }
298                     }
299                 {keyans*}
300                     {
301                         \tl_set:Nn \l__enumext_envir_name_tl { keyans* }
302                         \tl_set:Ne \l__enumext_check_start_line_env_tl
303                             {
304                                 in ~ 'keyans*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
305                             }
306                     }
307                 {keyanspic}
308                     {
309                         \tl_set:Nn \l__enumext_envir_name_tl { keyanspic }
310                         \tl_set:Ne \l__enumext_check_start_line_env_tl
311                             {
312                                 in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
313                             }
314                     }
315             }
316     }
```

(*End of definition for* __enumext_keyans_name_and_start:.)

12.5.2 Utilities for log and terminal

__enumext_reset_global_vars: The function __enumext_reset_global_vars: will be passed to the function __enumext_execute_-
__enumext_reset_global_int: after_env: and will return the global variables to their default values after being used.
__enumext_reset_global_bool:
__enumext_reset_global_tl:

```
317 \cs_new_protected:Nn \__enumext_reset_global_vars:
318     {
319         \__enumext_reset_global_int:
320         \__enumext_reset_global_bool:
321         \__enumext_reset_global_tl:
322     }
323 \cs_new_protected:Nn \__enumext_reset_global_int:
324     {
325         \int_gzero:N \g__enumext_item_number_int
326         \int_gzero:N \g__enumext_item_anskey_int
327         \int_gzero:N \g__enumext_item_answer_diff_int
328     }
329 \cs_new_protected:Nn \__enumext_reset_global_bool:
330     {
331         \bool_gset_false:N \g__enumext_check_ans_key_bool
332         \bool_gset_false:N \g__enumext_standar_bool
```

```
333     \bool_gset_false:N \g__enumext_starred_bool
334   }
335 \cs_new_protected:Nn \__enumext_reset_global_tl:
336   {
337     \tl_gclear:N \g__enumext_store_name_tl
338     \tl_gclear:N \g__enumext_start_line_tl
339     \tl_gclear:N \g__enumext_envir_name_tl
340   }
```

(*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.

```
341 \cs_new_protected:Nn \__enumext_log_global_vars:
342   {
343     \msg_log:nneeee { enumext } { prop-seq-int-hook }
344       { \g__enumext_store_name_tl }
345       { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
346       { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
347       { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
348   }
```

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.

```
349 \cs_new_protected:Nn \__enumext_log_answer_vars:
350   {
351     \msg_log:nneee { enumext } { item-answer-hook }
352       { \int_use:N \g__enumext_item_number_int }
353       { \int_use:N \g__enumext_item_anskey_int }
354       { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int} }
355   }
```

(*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.

```
356 \__enumext_at_begin_document:n
357   {
358     \cs_new_eq:NN \__enumext_start_list:nn \list
359     \cs_new_eq:NN \__enumext_stop_list: \endlist
360     \cs_new_eq:NN \__enumext_item_std:w \item
361   }
```

(*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.

```
362 \__enumext_at_begin_document:n
363   {
364     \cs_new_eq:NN \__enumext_minipage:w \minipage
365     \cs_new_eq:NN \__enumext_endminipage: \endminipage
366   }
```

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

12.7   The internal `minipage` environment

__enumext_internal_mini_page:
__enumext_mini_env*

The function `\__enumext_internal_mini_page:` creates a internal `__enumext_mini_env*` 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)

```
367  \cs_new_protected:Nn \__enumext_internal_mini_page:
368    {
369      \int_compare:nNnT { \l__enumext_level_int } = { 0 }
370        {
371          \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
372            {
373              \__enumext_minipage:w [ t ] { ##1 }
374                \legacy_if_gset_false:n { @minipage }
375                \skip_vertical:N \c_zero_skip
376            }
377            {
378                \skip_vertical:N \c_zero_skip
379              \__enumext_endminipage:
380            }
381        }
382    }
```

(*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.

```
383  \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
384  \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".

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

If the state of the variable `\l__enumext_footnotes_key_bool` is true we will check if the package `footnotehyper` is loaded, in case it is not present, we will set the value of `\l__enumext_footnotes_-key_bool` to false and we will redefine `\footnote`.

```
399      \bool_if:NT \l__enumext_footnotes_key_bool
400        {
401          \IfPackageLoadedTF { footnotehyper }
402            {
403              \msg_info:nnn { enumext } { package-load } { footnotehyper }
404            }
405            {
406              \typeout{No ~ footnotehyper ~ load}
407              \typeout{Load ~ and  ~ use  ~ \string\makesavenoteenv{enumext*}}
408              \bool_set_false:N \l__enumext_footnotes_key_bool
409            }
410        }
```

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.

```
411    \bool_if:NTF \l__enumext_hyperref_bool
412      {
413        \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
414        \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
415      }
416      {
417        \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
418        \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
419      }
420    }
```

(*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.

```
421  \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
422    {
423      \protected@write \@auxout { }
424        {
425          \token_to_str:N \newlabel {#1}
426            {
427              {#2}
428              \bool_if:NT \l__enumext_hyperref_bool
429                { { \thepage } {#2} {#1} }
430              { }
431            }
432        }
433      \__enumext_hypertarget:nn {#1} { }
434      \__enumext_phantomsection:
435    }
```

(*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.

```
436  \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.

```
437  \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
438    {
439      \cs_if_exist:cTF { c@ #2 }
440        { \msg_fatal:nnn { enumext } { counters }{ #2 } }
441        {
442          \tl_set:Nn #1 { #2 }
443          \newcounter { #2 }
444        }
445    }
```

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

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.

```
446  \__enumext_define_counters:Nn \l__enumext_counter_i_tl    { enumXi    }
447  \__enumext_define_counters:Nn \l__enumext_counter_ii_tl   { enumXii   }
448  \__enumext_define_counters:Nn \l__enumext_counter_iii_tl  { enumXiii  }
449  \__enumext_define_counters:Nn \l__enumext_counter_iv_tl   { enumXiv   }
450  \__enumext_define_counters:Nn \l__enumext_counter_v_tl    { enumXv    }
451  \__enumext_define_counters:Nn \l__enumext_counter_vi_tl   { enumXvi   }
452  \__enumext_define_counters:Nn \l__enumext_counter_vii_tl  { enumXvii  }
453  \__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.

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.

```
454  \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
455    {
456      \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
457      \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
458    }
459  \__enumext_register_counter_style:Nn \arabic { 0 }
460  \__enumext_register_counter_style:Nn \Alph   { M }
461  \__enumext_register_counter_style:Nn \alph   { m }
462  \__enumext_register_counter_style:Nn \Roman  { VIII }
463  \__enumext_register_counter_style:Nn \roman  { viii }
```

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

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

```
464  \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
465    {
466      \hbox_set:Nn \l__enumext_label_width_by_box {#2}
467      \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
468    }
469  \cs_generate_variant:Nn \__enumext_label_width_by_box:Nn { cv }
```

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

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

```
470  \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
471    {
472      \tl_clear_new:N #1
473      \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
474      \tl_gset_eq:NN \g__enumext_widest_label_tl #1
475      \tl_map_inline:Nn \g__enumext_counter_styles_tl
476        {
477          \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
478          \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
479            { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
480        }
481      \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
482        { \tl_use:N \g__enumext_widest_label_tl }
483      \tl_set_eq:cN { the #2 } #1
484    }
485  \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.

```
486 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
487   {
488     \keys_define:nn { enumext / #1 }
489       {
490         font        .tl_set:c   = { l__enumext_label_font_style_#2_tl },
491         font        .value_required:n = true,
492         labelsep    .dim_set:c  = { l__enumext_labelsep_#2_dim },
493         labelsep    .initial:n  = {0.3333em},
494         labelsep    .value_required:n = true,
495         labelwidth  .dim_set:c  = { l__enumext_labelwidth_#2_dim },
496         labelwidth  .value_required:n = true,
497         wrap-label  .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
498         wrap-label  .initial:n  = {##1},
499         wrap-label  .value_required:n = true,
500         wrap-label* .code:n = {
501                               \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
502                               \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
503                             },
504         wrap-label* .value_required:n = true,
505       }
506   }
507 \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.

```
508 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
509   {
510     \keys_define:nn { enumext / #1 }
511       {
512         align .choice:,
513         align / left    .code:n =
514                         {
515                           \tl_clear:c { l__enumext_label_fill_left_#2_tl  }
516                           \tl_set:cn  { l__enumext_label_fill_right_#2_tl } { \hfill }
517                         },
518         align / right   .code:n =
519                         {
520                           \tl_set:cn  { l__enumext_label_fill_left_#2_tl  } { \hfill }
521                           \tl_clear:c { l__enumext_label_fill_right_#2_tl }
522                         },
523         align / center  .code:n =
524                         {
525                           \tl_set:cn { l__enumext_label_fill_left_#2_tl  } { \hfill }
526                           \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
527                         },
528         align / unknown .code:n =
529                         \msg_error:nneee { enumext } { unknown-choice }
530                         { align } { left, ~ right, ~  center } { \exp_not:n {##1} },
531         align .initial:n  = left,
532         align .value_required:n  = true,
533       }
534   }
535 \clist_map_inline:nn
536   {
537     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
538   }
539   { \__enumext_tmp:nn #1 }

540 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
541   {
542     \keys_define:nn { enumext / #1 }
543       {
544         align .choice:,
```

```
545        align / left     .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
546        align / right    .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
547        align / center   .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
548        align / unknown .code:n =
549                             \msg_error:nneee { enumext } { unknown-choice }
550                               { align } { left, ~ right, ~  center } { \exp_not:n {##1} },
551        align .initial:n = left,
552        align .value_required:n = true,
553      }
554    }
555 \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.

```
556 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
557   {
558     \keys_define:nn { enumext / #1 }
559       {
560         label .code:n     = {
561                             \__enumext_label_style:cvn { l__enumext_label_#2_tl }
562                               { l__enumext_counter_#2_tl } {##1}
563                             \dim_set_eq:cN  { l__enumext_labelwidth_#2_dim }
564                               \l__enumext_current_widest_dim
565                           },
566         label .initial:n = #3,
567         label .value_required:n = true,
568         ref   .code:n     = \__enumext_standar_ref:n {##1},
569         ref   .value_required:n = true,
570       }
571   }
572 \__enumext_tmp:nnn { level-1 } {   i } { \arabic*.}
573 \__enumext_tmp:nnn { level-2 } {  ii } { (\alph*) }
574 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
575 \__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.

```
576 \cs_new_protected:Npn \__enumext_standar_ref:n #1
577   {
578     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
579     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
580       {
581         \msg_error:nnn { enumext } { key-ref-empty } { enumext }
582       }
583       {
584         \tl_set_eq:Nc
585           \l__enumext_ref_the_count_tl { l__enumext_counter_ \__enumext_level: _tl }
586         \__enumext_regex_counter_style:
587         \tl_set_eq:Nc
588           \l__enumext_ref_the_count_tl { l__enumext_the_counter_ \__enumext_level: _tl }
589         \tl_put_right:ce { l__enumext_renew_the_count_ \__enumext_level: _tl }
590           {
591             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
592               { \exp_not:V \l__enumext_ref_key_arg_tl }
593           }
594       }
595   }
```

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

```
596  \cs_new_protected:Nn \__enumext_standar_ref:
597    {
598      \tl_if_empty:cF { l__enumext_renew_the_count_ \__enumext_level: _tl }
599        {
600          \tl_use:c { l__enumext_renew_the_count_ \__enumext_level: _tl }
601        }
602    }
```

(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.

```
603  \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
604    {
605      \keys_define:nn { enumext / #1 }
606        {
607          label .code:n    = {
608                             \__enumext_label_style:cvn { l__enumext_label_#2_tl }
609                               { l__enumext_counter_#2_tl } {##1}
610                             \dim_set_eq:cN  { l__enumext_labelwidth_#2_dim }
611                               \l__enumext_current_widest_dim
612                           },
613          label .initial:n = #3,
614          label .value_required:n = true,
615          ref    .code:n    = \__enumext_starred_ref:n {##1},
616          ref    .value_required:n = true,
617        }
618    }
619  \__enumext_tmp:nnn { enumext* } {  vii } { \arabic*.}
620  \__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`.

```
621  \cs_new_protected:Npn \__enumext_starred_ref:n #1
622    {
623      \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
624      \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
625        {
626          \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
627            {
628              \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
629            }
630            {
631              \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
632              \__enumext_regex_counter_style:
633              \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
634              \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
635                {
636                  \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
637                    { \exp_not:V \l__enumext_ref_key_arg_tl }
638                }
639            }
640        }
641      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
642        {
643          \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
644            {
645              \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
646            }
647            {
648              \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
649              \__enumext_regex_counter_style:
650              \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
651              \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
652                {
653                  \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
```

```
654                      { \exp_not:V \l__enumext_ref_key_arg_tl }
655                  }
656              }
657          }
658      }
```

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.

```
659 \cs_new_protected:Nn \__enumext_starred_ref:
660   {
661     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
662       {
663         \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
664           {
665             \tl_use:N \l__enumext_renew_the_count_vii_tl
666           }
667       }
668     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
669       {
670         \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
671           {
672             \tl_use:N \l__enumext_renew_the_count_viii_tl
673           }
674       }
675   }
```

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

12.13.3 Define and set `label` and `ref` keys for `keyans` and `keyanspic` environments

label
ref
`\l__enumext_label_v_tl`
`\l__enumext_label_vi_tl`

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

```
676 \keys_define:nn { enumext / keyans }
677   {
678     label .code:n    = {
679                        \__enumext_label_style:cvn { l__enumext_label_v_tl }
680                          { l__enumext_counter_v_tl } {#1}
681                        \dim_set_eq:cN  { l__enumext_labelwidth_v_dim }
682                          \l__enumext_current_widest_dim
683                        \__enumext_label_style:cvn { l__enumext_label_vi_tl }
684                          { l__enumext_counter_vi_tl } {#1}
685                        \dim_set_eq:cN  { l__enumext_labelwidth_v_dim }
686                          \l__enumext_current_widest_dim
687                      },
688     label .initial:n = \Alph*),
689     label .value_required:n = true,
690     ref   .code:n    = \__enumext_keyans_ref:n {#1},
691     ref   .value_required:n = true,
692   }
```

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

`\__enumext_keyans_ref:n`
`\__enumext_keyans_ref:`

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

```
693 \cs_new_protected:Npn \__enumext_keyans_ref:n #1
694   {
695     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
696     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
697       {
698         \msg_error:nnn { enumext } { key-ref-empty } { keyans }
699       }
700       {
701         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
702         \__enumext_regex_counter_style:
703         \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
704         \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
705           {
706             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
707               { \exp_not:V \l__enumext_ref_key_arg_tl }
708           }
709       }
710   }
```

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

```
711  \cs_new_protected:Nn \__enumext_keyans_ref:
712    {
713      \tl_if_empty:NF \l__enumext_renew_the_count_v_tl
714        {
715          \tl_use:N \l__enumext_renew_the_count_v_tl
716        }
717    }
```

(*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.

```
718  \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
719    {
720      \__enumext_if_is_int:nTF { #3 }
721        {
722          \int_set:Nn #2 {#3}
723        }
724        {
725          \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
726            { \int_set:Nn #2 { \int_from_alph:n {#3} } }
727          \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
728            { \int_set:Nn #2  { \int_from_roman:n {#3} } }
729        }
730    }
731  \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.

```
732  \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
733    {
734      \__enumext_if_is_int:nTF {#4}
735        {
736          \setcounter{enumX#1} { #4 }
737        }
738        {
739          \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
740            { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
741          \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
742            { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
743        }
744      \__enumext_label_width_by_box:cv
745        { l__enumext_labelwidth_#1_dim } { l__enumext_label_#1_tl }
746    }
747  \cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }
```

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

start
start*
widest

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

```
748  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
749    {
```

```
750    \keys_define:nn { enumext / #1 }
751      {
752        start* .code:n    = {
753                            \__enumext_start_from:ccn
754                              { l__enumext_label_#2_tl }
755                              { l__enumext_start_#2_int } {##1}
756                            },
757        start* .value_required:n = true,
758        start  .code:n    = {
759                            \__enumext_start_from:cce
760                              { l__enumext_label_#2_tl }
761                              { l__enumext_start_#2_int } { \int_eval:n {##1} }
762                            },
763        start  .initial:n = 1,
764        start  .value_required:n = true,
765        widest .code:n    = {
766                            \__enumext_widest_from:nccn {#2}
767                              { l__enumext_label_#2_tl }
768                              { l__enumext_labelwidth_#2_dim } {##1}
769                            },
770        widest .value_required:n = true,
771      }
772    }
773 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

12.15   Setting keys for vertical spaces

topsep
partopsep
parsep
noitemsep
nosep

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

```
774 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
775   {
776    \keys_define:nn { enumext / #1 }
777      {
778        topsep    .skip_set:c = { l__enumext_topsep_#2_skip },
779        topsep    .initial:n  = {#3},
780        topsep    .value_required:n = true,
781        partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
782        partopsep .initial:n  = {#4},
783        partopsep .value_required:n = true,
784        parsep    .skip_set:c = { l__enumext_parsep_#2_skip },
785        parsep    .initial:n  = {#5},
786        parsep    .value_required:n = true,
787        itemsep   .skip_set:c = { l__enumext_itemsep_#2_skip },
788        itemsep   .initial:n  = {#6},
789        itemsep   .value_required:n = true,
790        noitemsep .meta:n    = { itemsep = 0pt, parsep = 0pt },
791        noitemsep .value_forbidden:n = true,
792        nosep     .meta:n    = {
793                              itemsep = 0pt, parsep= 0pt,
794                              topsep = 0pt, partopsep = 0pt,
795                              },
796        nosep     .value_forbidden:n = true,
797      }
798    }
```

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

```
799 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
800   { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
801   { 4.0pt plus 2.0pt minus 1.0pt }
802 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
803   { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
804   { 2.0pt plus 1.0pt minus 1.0pt }
805 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
806   { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
807 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
808   { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
809 \__enumext_tmp:nnnnnn { keyans  } { v }{ 4.0pt plus 2.0pt minus 1.0pt }
810   { 2.0pt plus 1.0pt minus 1.0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
811   { 2.0pt plus 1.0pt minus 1.0pt }
812 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
```

```
813    { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
814    { 4.0pt plus 2.0pt minus 1.0pt }
815 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
816    { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
817    { 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   We define the key base-fix only for the *"first level"* of enumext and enumext*.
__enumext_nested_base_line_fix:

```
818 \cs_set_protected:Npn \__enumext_tmp:n #1
819    {
820      \keys_define:nn { enumext / #1 }
821        {
822          base-fix .bool_set:N = \l__enumext_base_line_fix_bool,
823          base-fix .initial:n  = false,
824          base-fix .value_forbidden:n = true,
825        }
826    }
827 \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).

```
828 \cs_new_protected:Nn \__enumext_nested_base_line_fix:
829    {
830      \bool_lazy_and:nnT
831        { \bool_if_p:N \l__enumext_standar_first_bool }
832        { \bool_if_p:N \l__enumext_base_line_fix_bool }
833        {
834          \mode_leave_vertical:
835          \vspace { -\baselineskip }
836          \keys_set:nn { enumext / level-1 }
837            {
838              topsep = 0pt, above = 0pt, above* = 0pt,
839            }
840        }
841      \bool_lazy_and:nnT
842        { \bool_if_p:N \l__enumext_starred_first_bool }
843        { \bool_if_p:N \l__enumext_base_line_fix_bool }
844        {
845          \mode_leave_vertical:
846          \vspace { -\baselineskip }
847          \keys_set:nn { enumext / enumext* }
848            {
849              topsep = 0pt, above = 0pt, above* = 0pt,
850            }
851        }
852      \bool_set_false:N \l__enumext_base_line_fix_bool
853    }
```

🎯 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   Define and set itemindent, rightmargin, listparindent, list-offset and list-indent keys for
rightmargin   enumext, enumext*, keyans and keyans* environments.
listparindent
list-offset
list-indent
```
854 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
855    {
856      \keys_define:nn { enumext / #1 }
857        {
858          itemindent    .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
859          itemindent    .value_required:n = true,
860          rightmargin   .dim_set:c = { l__enumext_rightmargin_#2_dim },
```

```
861        rightmargin   .value_required:n = true,
862        listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
863        listparindent .value_required:n = true,
864        list-offset   .dim_set:c = { l__enumext_listoffset_#2_dim },
865        list-offset   .value_required:n = true,
866        list-indent   .code:n    =
867                        \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
868                        \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
869        list-indent   .value_required:n = true,
870      }
871    }
872  \clist_map_inline:nn
873    {
874      {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
875    }
876    { \__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.

```
877  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
878    {
879      \keys_define:nn { enumext / #1 }
880        {
881          itemindent    .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
882          itemindent    .value_required:n = true,
883          rightmargin   .dim_set:c = { l__enumext_rightmargin_#2_dim },
884          rightmargin   .value_required:n = true,
885          listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
886          listparindent .value_required:n = true,
887          list-offset   .dim_set:c = { l__enumext_listoffset_#2_dim },
888          list-offset   .value_required:n = true,
889          list-indent   .meta:n     = { list-offset = ##1 },
890          list-indent   .value_required:n = true,
891        }
892    }
893  \clist_map_inline:nn
894    {
895      {enumext*}{vii}, {keyans*}{viii}
896    }
897    { \__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.

```
898  \cs_set_protected:Nn \__enumext_fake_item:
899    {
900      \dim_compare:nNnT
901        { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
902        >
903        { \c_zero_dim }
904        {
905          \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
906            {
907              \exp_not:N \mode_leave_vertical:
908              \exp_not:n { \skip_horizontal:n }
909                { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
910              \ignorespaces
911            }
912        }
913    }
914  \cs_set_protected:Nn \__enumext_keyans_fake_item:
915    {
916      \dim_compare:nNnT
917        { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
918        {
919          \tl_set:Ne \l__enumext_fake_item_indent_v_tl
920            {
```

```
921            \exp_not:N \mode_leave_vertical:
922            \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
923          }
924        }
925      }
926  \cs_set_protected:Nn \__enumext_fake_item_vii:
927    {
928      \dim_compare:nNnT
929        { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
930        {
931          \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
932            {
933              \exp_not:N \mode_leave_vertical:
934              \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
935            }
936        }
937    }
938  \cs_set_protected:Nn \__enumext_fake_item_viii:
939    {
940      \dim_compare:nNnT
941        { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
942        {
943          \tl_set:Ne \l__enumext_fake_item_indent_viii_tl
944            {
945              \exp_not:N \mode_leave_vertical:
946              \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
947            }
948        }
949    }
```

(*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.

```
950  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
951    {
952      \keys_define:nn { enumext / #1 }
953        {
954          show-length .bool_set:c = { l__enumext_show_length_#2_bool },
955          show-length .initial:n  = false,
956        }
957    }
958  \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
```
959  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
960    {
961      \keys_define:nn { enumext / #1 }
962        {
963          before  .tl_set:c   = { l__enumext_before_no_starred_key_#2_tl },
964          before  .value_required:n = true,
965          before* .tl_set:c   = { l__enumext_before_starred_key_#2_tl },
966          before* .value_required:n = true,
967          after   .tl_set:c   = { l__enumext_after_stop_list_#2_tl },
968          after   .value_required:n = true,
969          first   .tl_set:c   = { l__enumext_after_list_args_#2_tl },
970          first   .value_required:n = true,
971        }
972    }
973  \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*⟩}.

```
974 \cs_new_protected:Nn \__enumext_before_args_exec:
975   {
976     \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
977   }
```

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*⟩}}

```
978 \cs_new_protected:Nn \__enumext_before_keys_exec:
979   {
980     \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
981   }
```

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

```
982 \cs_new_protected:Nn \__enumext_after_stop_list:
983   {
984     \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
985   }
```

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.

```
986 \cs_new_protected:Nn \__enumext_after_args_exec:
987   {
988     \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
989   }
```

(*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.

```
990 \cs_new_protected:Nn \__enumext_before_args_exec_v:
991   {
992     \tl_use:N \l__enumext_before_starred_key_v_tl
993   }
994 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
995   {
996     \tl_use:N \l__enumext_before_no_starred_key_v_tl
997   }
998 \cs_new_protected:Nn \__enumext_after_stop_list_v:
999   {
1000     \tl_use:N \l__enumext_after_stop_list_v_tl
1001   }
1002 \cs_new_protected:Nn \__enumext_after_args_exec_v:
1003   {
1004     \tl_use:N \l__enumext_after_list_args_v_tl
1005   }
```

(*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.

```
1006 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
1007   {
1008     \tl_use:N \l__enumext_before_starred_key_vii_tl
1009   }
1010 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
1011   {
1012     \tl_use:N \l__enumext_before_starred_key_viii_tl
1013   }
1014 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
1015   {
1016     \tl_use:N \l__enumext_before_no_starred_key_vii_tl
1017   }
1018 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
1019   {
```

```
1020        \tl_use:N \l__enumext_before_no_starred_key_viii_tl
1021     }
1022  \cs_new_protected:Nn \__enumext_after_stop_list_vii:
1023     {
1024        \tl_use:N \l__enumext_after_stop_list_vii_tl
1025     }
1026  \cs_new_protected:Nn \__enumext_after_stop_list_viii:
1027     {
1028        \tl_use:N \l__enumext_after_stop_list_viii_tl
1029     }
1030  \cs_new_protected:Nn \__enumext_after_args_exec_vii:
1031     {
1032        \tl_use:N \l__enumext_after_list_args_vii_tl
1033     }
1034  \cs_new_protected:Nn \__enumext_after_args_exec_viii:
1035     {
1036        \tl_use:N \l__enumext_after_list_args_viii_tl
1037     }
```

(*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.

```
1038  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1039     {
1040        \keys_define:nn { enumext / #1 }
1041          {
1042            mini-env     .dim_set:c  = { l__enumext_minipage_right_#2_dim },
1043            mini-env     .value_required:n = true,
1044            mini-sep     .dim_set:c  = { l__enumext_minipage_hsep_#2_dim },
1045            mini-sep     .initial:n  = 0.3333em,
1046            mini-sep     .value_required:n = true,
1047            columns-sep .dim_set:c  = { l__enumext_columns_sep_#2_dim },
1048            columns-sep .value_required:n = true,
1049            columns      .int_set:c  = { l__enumext_columns_#2_int },
1050            columns      .initial:n  = 1,
1051            columns      .value_required:n = true,
1052          }
1053     }
1054  \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.

```
1055  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1056     {
1057        \keys_define:nn { enumext / #1 }
1058          {
1059            mini-right  .tl_gset:c = { g__enumext_miniright_code_#2_tl },
1060            mini-right  .value_required:n = true,
1061            mini-right* .code:n    = {
1062                                     \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
1063                                     \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1064                                   },
1065            mini-right* .value_required:n = true,
1066          }
1067     }
1068  \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.

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`.

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

🎯 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*.

```
1069  \cs_new_protected:Nn \__enumext_multi_set_vskip:
1070    {
1071      \skip_set:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1072        {
1073          \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1074        }
1075      \skip_set:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1076        {
1077          \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1078        }
1079      \__enumext_add_pre_parsep:
1080    }
```

(*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*.

```
1081  \cs_new_protected:Nn \__enumext_add_pre_parsep:
1082    {
1083      \int_case:nn { \l__enumext_level_int }
1084        {
1085          { 2 }{
1086              \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1087                {
1088                  \skip_add:Nn \l__enumext_multicols_above_ii_skip { \l__enumext_parsep_i_skip }
1089                }
1090            }
1091          { 3 }{
1092              \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1093                {
1094                  \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_ii_skip
1095                }
1096            }
1097          { 4 }{
1098              \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1099                {
1100                  \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip
1101                }
1102            }
1103        }
1104    }
```

(*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*⟩.

```
1105 \cs_new_protected:Nn \__enumext_multi_addvspace:
1106   {
1107     \__enumext_multi_set_vskip:
1108     \mode_if_vertical:T
1109       {
1110         \skip_add:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1111           {
1112             \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1113           }
1114         \skip_add:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1115           {
1116             \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1117           }
1118       }
1119     \par\nopagebreak
1120     \addvspace{ \skip_use:c { l__enumext_multicols_above_ \__enumext_level: _skip } }
1121   }
```

(*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.

```
1122 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1123   {
1124     \skip_set:Nn \l__enumext_multicols_above_v_skip
1125       {
1126         \l__enumext_topsep_v_skip
1127       }
1128     \skip_set:Nn \l__enumext_multicols_below_v_skip
1129       {
1130         \l__enumext_topsep_v_skip
1131       }
1132   }
1133 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1134   {
1135     \__enumext_keyans_multi_set_vskip:
1136     \mode_if_vertical:T
1137       {
1138         \skip_add:Nn \l__enumext_multicols_above_v_skip
1139           {
1140             \skip_use:N \l__enumext_partopsep_v_skip
1141           }
1142         \skip_add:Nn \l__enumext_multicols_below_v_skip
1143           {
1144             \skip_use:N \l__enumext_partopsep_v_skip
1145           }
1146       }
1147     \par\nopagebreak
1148     \addvspace{ \l__enumext_multicols_above_v_skip }
1149   }
```

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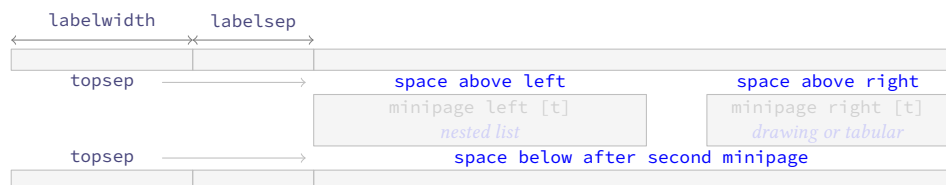
Figure 8: Representation of the minipage spacing adjustment for a nested level.

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_mini_set_vskip:

The function __enumext_mini_set_vskip: will take care of determining the *"adjust"* spaces that we will apply *"above"* and *"below"* the __enumext_mini_env* 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.

First determine if the multicols environment is active by comparing the value of the \l__enumext_-columns_X_int variable handled by the columns key, according to this comparison we set the adjusted values for \l__enumext_minipage_left_skip, \l__enumext_minipage_right_skip and \l__-enumext_minipage_after_skip.

```
1150  \cs_new_protected:Nn \__enumext_mini_set_vskip:
1151    {
1152      \int_compare:nNnTF
1153        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1154        {
```

If multicols environment is nested in __enumext_mini_env* environment, we will apply a correction factor to the *vertical spaces* taking into account the value of \topsep of the current level and the value of \parsep of the previous level, if these are zero we will use \strutbox as the basis for the calculations.

```
1155          \skip_if_eq:nnTF
1156            { \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1157            {
1158              \skip_set:Nn \l__enumext_minipage_left_skip
1159                {
1160                  -0.150\box_dp:N \strutbox
1161                }
1162              \skip_set:Nn \l__enumext_minipage_right_skip
1163                {
1164                  0.695\box_dp:N \strutbox
1165                }
1166              \skip_set:Nn \l__enumext_minipage_after_skip
1167                {
1168                  \box_dp:N \strutbox
1169                }
1170              \__enumext_zero_parsep:
1171            }
1172            {
1173              \skip_set:Nn \l__enumext_minipage_left_skip
1174                {
1175                  \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1176                }
1177              \skip_set:Nn \l__enumext_minipage_right_skip
1178                {
1179                  0.695\box_dp:N \strutbox
1180                }
1181              \skip_set:Nn \l__enumext_minipage_after_skip
1182                {
1183                  1.85\box_dp:N \strutbox
1184                  + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1185                }
1186            }
1187        }
1188        {
```

If only enumext environment is nested in __enumext_mini_env* environment, we will apply a correction factor to the *vertical spaces* taking into account the value of \topsep, if this is zero we will use \strutbox as the basis for the calculations.

```
1189            \skip_if_eq:nnTF
1190              { \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1191              {
1192                \skip_set:Nn \l__enumext_minipage_left_skip
1193                  {
1194                    0.5\box_dp:N \strutbox
1195                    - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1196                  }
1197                \skip_set:Nn \l__enumext_minipage_right_skip
1198                  {
1199                    \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1200                  }
1201                \skip_set:Nn \l__enumext_minipage_after_skip
1202                  {
1203                    1.6\box_dp:N \strutbox
1204                  }
1205              }
1206              {
1207                \skip_set:Nn \l__enumext_minipage_left_skip
1208                  {
1209                    0.5875\box_dp:N \strutbox
1210                    - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1211                  }
1212                \skip_set:Nn \l__enumext_minipage_right_skip
1213                  {
1214                    + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1215                    + \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1216                  }
1217                \skip_set:Nn \l__enumext_minipage_after_skip
1218                  {
1219                    0.325\box_dp:N \strutbox
1220                    + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1221                  }
1222              }
1223          }
1224      }
```

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

`\__enumext_zero_parsep:`   The function `\__enumext_zero_parsep:` *"adjusted"* the value of `\l__enumext_minipage_after_-skip` detecting the value of `\parsep` from the previous level. This is necessary since `\parsep` from the previous level affects the *vertical spaces* and this is noticeable when using the `nosep` or `noitemsep` keys.

```
1225  \cs_new_protected:Nn \__enumext_zero_parsep:
1226    {
1227      \int_case:nn { \l__enumext_level_int }
1228        {
1229          { 2 }{
1230                \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1231                  {
1232                    \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1233                  }
1234              }
1235          { 3 }{
1236                \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1237                  {
1238                    \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1239                  }
1240              }
1241          { 4 }{
1242                \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1243                  {
1244                    \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1245                  }
1246              }
1247        }
1248    }
```

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

`\__enumext_mini_addvspace:`   The function `\__enumext_mini_addvspace:` will apply the spaces set using `\addvspace` *"above"* the `__enumext_mini_env*` environment in `enumext`, taking into account whether TeX is in ⟨*horizontal mode*⟩

or ⟨*vertical mode*⟩. For the latter we will make some adjustments since the `\partopsep` parameter comes into play and this affects the *vertical spacing*.

```
1249 \cs_new_protected:Nn \__enumext_mini_addvspace:
1250   {
1251     \__enumext_mini_set_vskip:
1252     \mode_if_vertical:T
1253       {
1254         \skip_add:Nn \l__enumext_minipage_left_skip
1255           {
1256             \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1257           }
1258         \skip_add:Nn \l__enumext_minipage_after_skip
1259           {
1260             \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1261           }
1262       }
1263     \par\nopagebreak
1264     \addvspace { \l__enumext_minipage_left_skip }
1265   }
```

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

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

`\__enumext_keyans_mini_set_vskip:` 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_env*` environment in `keyans`. The implementation of this function is the same as the one used in `enumext`.

```
1266 \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1267   {
1268     \skip_zero_new:N \l__enumext_minipage_after_skip
1269     \skip_zero_new:N \l__enumext_minipage_left_skip
1270     \skip_zero_new:N \l__enumext_minipage_right_skip
1271     \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1272       {
1273         \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1274           {
1275             \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1276             \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1277             \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1278             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1279               {
1280                 \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1281               }
1282           }
1283           {
1284             \skip_set:Nn \l__enumext_minipage_left_skip
1285               {
1286                 \skip_use:N \l__enumext_topsep_v_skip
1287               }
1288             \skip_set:Nn \l__enumext_minipage_right_skip
1289               {
1290                 0.705\box_dp:N \strutbox
1291               }
1292             \skip_set:Nn \l__enumext_minipage_after_skip
1293               {
1294                 1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1295               }
1296           }
1297       }
1298       {
1299         \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1300           {
1301             \skip_set:Nn \l__enumext_minipage_left_skip
1302               {
1303                 0.5\box_dp:N \strutbox
1304                 + \l__enumext_partopsep_v_skip
1305               }
1306             \skip_set:Nn \l__enumext_minipage_right_skip
1307               {
1308                 \l__enumext_partopsep_v_skip
1309               }
```

```
1310        \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1311        }
1312      {
1313        \skip_set:Nn \l__enumext_minipage_left_skip
1314          {
1315            0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1316          }
1317        \skip_set:Nn \l__enumext_minipage_right_skip
1318          {
1319            \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1320          }
1321        \skip_set:Nn \l__enumext_minipage_after_skip
1322          {
1323            0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1324          }
1325      }
1326    }
1327  }
```

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

`\__enumext_keyans_mini_addvspace:`   The function `\__enumext_keyans_mini_addvspace:` will apply the spaces set using `\addvspace` *"above"* the `__enumext_mini_env*` environment in `keyans`, taking into account whether TEX is in ⟨*horizontal mode*⟩ or ⟨*vertical mode*⟩. For the latter we will make some adjustments since the `\partopsep` parameter comes into play and this affects the *vertical spacing*. The implementation of this function is the same as the one used in `enumext`.

```
1328  \cs_new_protected:Nn \__enumext_keyans_mini_addvspace:
1329    {
1330      \__enumext_keyans_mini_set_vskip:
1331      \mode_if_vertical:T
1332        {
1333          \skip_add:Nn \l__enumext_minipage_left_skip
1334            {
1335              \l__enumext_partopsep_v_skip
1336            }
1337          \skip_add:Nn \l__enumext_minipage_after_skip
1338            {
1339              \l__enumext_partopsep_v_skip
1340            }
1341        }
1342      \par\nopagebreak
1343      \addvspace { \l__enumext_minipage_left_skip }
1344    }
```

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

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_env*` environment in `enumext*` and `keyans*`.

```
1345  \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1346    {
1347      \skip_zero_new:N \l__enumext_minipage_left_skip
1348      \skip_gzero_new:N \g__enumext_minipage_right_skip
1349      \skip_gzero_new:N \g__enumext_minipage_after_skip
1350      \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1351        {
1352          \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1353          \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1354        }
1355        {
1356          \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1357          \skip_gset:Nn \g__enumext_minipage_right_skip
1358            {
1359              \l__enumext_topsep_vii_skip
1360            }
1361          \skip_gset:Nn \g__enumext_minipage_after_skip
1362            {
1363              0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1364            }
```

```
1365              }
1366          }
1367    \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1368      {
1369        \skip_zero_new:N \l__enumext_minipage_after_skip
1370        \skip_zero_new:N \l__enumext_minipage_left_skip
1371        \skip_zero_new:N \l__enumext_minipage_right_skip
1372        \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1373          {
1374            \skip_set:Nn \l__enumext_minipage_left_skip
1375              {
1376                0.5\box_dp:N \strutbox
1377              }
1378            \skip_set:Nn \l__enumext_minipage_right_skip
1379              {
1380                \l__enumext_partopsep_viii_skip
1381              }
1382            \skip_set:Nn \l__enumext_minipage_after_skip
1383              {
1384                1.6\box_dp:N \strutbox
1385              }
1386          }
1387          {
1388            \skip_set:Nn \l__enumext_minipage_left_skip
1389              {
1390                0.5875\box_dp:N \strutbox
1391              }
1392            \skip_set:Nn \l__enumext_minipage_right_skip
1393              {
1394                \l__enumext_topsep_viii_skip
1395              }
1396            \skip_set:Nn \l__enumext_minipage_after_skip
1397              {
1398                0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1399              }
1400          }
1401      }
```

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

__enumext_mini_addvspace_vii:
__enumext_mini_addvspace_viii:

The functions __enumext_mini_addvspace_vii: and __enumext_mini_addvspace_viii: will apply the vertical space *"only above"* the __enumext_mini_env* 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.

```
1402    \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1403      {
1404        \__enumext_mini_set_vskip_vii:
1405        \par\nopagebreak
1406        \addvspace { \l__enumext_minipage_left_skip }
1407      }
1408    \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1409      {
1410        \__enumext_mini_set_vskip_viii:
1411        \par\nopagebreak
1412        \addvspace { \l__enumext_minipage_left_skip }
1413      }
```

(*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_env* environment on the *"left side"*, open the __enumext_mini_env* 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_env* on the *"right side"*.

\miniright

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

```
1414  \NewDocumentCommand \miniright { s }
1415    {
1416      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1417        {
1418          \msg_error:nnn { enumext } { wrong-miniright-place }
1419        }
1420      % outside
1421      \bool_lazy_and:nnT
1422        { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
1423        { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
1424        {
1425          \msg_error:nnn { enumext } { wrong-miniright-place }
1426        }
1427      % starred env
1428      \bool_if:NT \l__enumext_starred_bool
1429        {
1430          \msg_error:nnn { enumext } { wrong-miniright-starred }
1431        }
1432      \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
1433        {
1434          \__enumext_keyans_mini_right_cmd:n {#1}
1435        }
1436        { \__enumext_mini_right_cmd:n {#1} }
1437    }
```

(*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_env* environment on the *"left side"*, then we open the __enumext_mini_env* 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.

```
1438  \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1439    {
1440      \dim_compare:nNnTF
1441        { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1442        {
1443          \__enumext_multicols_stop:
1444          \end{__enumext_mini_env*}
1445          \hfill
1446          \begin{__enumext_mini_env*}
1447            { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } }
1448            \par\addvspace { \l__enumext_minipage_right_skip }
1449            \bool_if:nF {#1}
1450              {
1451                \centering
1452              }
1453            \int_gzero:N \g__enumext_minipage_stat_int
1454        }
1455        { \msg_error:nnn { enumext } { wrong-miniright-use } }
1456      % paranoia
1457      \RenewDocumentCommand \miniright { s }
1458        {
1459          \msg_error:nn { enumext } { many-miniright-used }
1460        }
1461    }
```

(*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.

```
1462  \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1463    {
1464      \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1465        {
1466          \__enumext_keyans_multicols_stop:
1467          \end{__enumext_mini_env*}
1468          \hfill
```

```
1469        \begin{__enumext_mini_env*}{ \l__enumext_minipage_right_v_dim }
1470          \par\addvspace { \l__enumext_minipage_right_skip }
1471          \bool_if:nF {#1}
1472            {
1473              \centering
1474            }
1475          \int_gzero:N \g__enumext_minipage_stat_int
1476        }
1477      { \msg_error:nnn { enumext } { wrong-miniright-use } }
1478    \RenewDocumentCommand \miniright { s }
1479      {
1480        \msg_error:nn { enumext } { many-miniright-used }
1481      }
1482    }
```

(*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.

```
1483 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1484   {
1485     \keys_define:nn { enumext / #1 }
1486       {
1487         above  .skip_set:c = { l__enumext_vspace_above_#2_skip },
1488         above  .value_required:n = true,
1489         above* .code:n     = \bool_set_true:c { l__enumext_vspace_a_star_#2_bool }
1490                              \keys_set:nn { enumext / #1 } { above = {##1} },
1491         above* .value_required:n = true,
1492         below  .skip_set:c = { l__enumext_vspace_below_#2_skip },
1493         below  .value_required:n = true,
1494         below* .code:n     = \bool_set_true:c { l__enumext_vspace_b_star_#2_bool }
1495                              \keys_set:nn { enumext / #1 } { below = {##1} },
1496         below* .value_required:n = true,
1497       }
1498   }
1499 \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.

```
1500 \cs_new_protected:Nn \__enumext_vspace_above:
1501   {
1502     \skip_if_eq:nnF
1503       { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1504       {
1505         \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
1506           {
1507             \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1508           }
1509           {
1510             \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1511           }
1512       }
1513   }
```

(*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.

```
1514 \cs_new_protected:Nn \__enumext_vspace_below:
1515   {
1516     \skip_if_eq:nnF
```

```
1517        { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1518        {
1519          \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1520            {
1521              \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1522            }
1523            {
1524              \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1525            }
1526        }
1527    }
```

(*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.

```
1528 \cs_new_protected:Nn \__enumext_vspace_above_v:
1529    {
1530      \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1531        {
1532          \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1533            {
1534              \vspace*{ \l__enumext_vspace_above_v_skip }
1535            }
1536            { \vspace { \l__enumext_vspace_above_v_skip } }
1537        }
1538    }
```

(*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.

```
1539 \cs_new_protected:Nn \__enumext_vspace_below_v:
1540    {
1541      \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1542        {
1543          \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1544            {
1545              \vspace*{ \l__enumext_vspace_below_v_skip }
1546            }
1547            { \vspace { \l__enumext_vspace_below_v_skip } }
1548        }
1549    }
```

(*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
__enumext_vspace_above_viii: *vertical space above* the enumext* and keyans* environments set by the above and above* keys.

```
1550 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1551    {
1552      \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1553        {
1554          \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1555            {
1556              \vspace*{ \l__enumext_vspace_above_vii_skip }
1557            }
1558            { \vspace { \l__enumext_vspace_above_vii_skip } }
1559        }
1560    }
1561 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1562    {
1563      \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1564        {
1565          \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1566            {
1567              \vspace*{ \l__enumext_vspace_above_viii_skip }
1568            }
1569            { \vspace { \l__enumext_vspace_above_viii_skip } }
1570        }
1571    }
```

(*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.

```
1572 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1573   {
1574     \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1575       {
1576         \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1577           {
1578             \vspace*{ \l__enumext_vspace_below_vii_skip }
1579           }
1580           { \vspace { \l__enumext_vspace_below_vii_skip } }
1581       }
1582   }
1583 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1584   {
1585     \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1586       {
1587         \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1588           {
1589             \vspace*{ \l__enumext_vspace_below_viii_skip }
1590           }
1591           { \vspace { \l__enumext_vspace_below_viii_skip } }
1592       }
1593   }
```

(*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*.

```
1594 \cs_set_protected:Npn \__enumext_tmp:n #1
1595   {
1596     \keys_define:nn { enumext / #1 }
1597       {
1598         series  .str_set:N = \l__enumext_series_str,
1599         series  .value_required:n = true,
1600         resume  .code:n = \__enumext_resume_series:n {##1},
1601         resume* .code:n = \__enumext_resume_starred:,
1602         resume* .value_forbidden:n = true,
1603       }
1604   }
1605 \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.

```
1606 \cs_new:Npn \__enumext_filter_series:n #1
1607   {
1608     \use:e
1609       {
1610         \keyval_parse:NNn
1611           \__enumext_filter_series_key:n
1612           \__enumext_filter_series_pair:nn {#1}
1613       }
1614   }
```

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.

```
1615 \cs_new:Npn \__enumext_filter_series_key:n #1
1616   {
1617     \str_case:nnF {#1}
1618       {
1619         { resume } {} { resume* } {} { base-fix } {}
```

```
1620        }
1621        { , { \exp_not:n {#1} } }
1622    }
```

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.

```
1623 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1624    {
1625      \str_case:nnF {#1}
1626        {
1627          { series } {} { resume } {} { start } {}
1628          { start* } {}  { save-ans } {} { save-key } {}
1629        }
1630        { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1631    }
```

(*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).

```
1632 \cs_new_protected:Npn \__enumext_parse_series:n #1
1633    {
1634      \str_if_empty:NTF \l__enumext_series_str
1635        {
1636          \bool_if:NF \l__enumext_resume_active_bool
1637            {
1638              \__enumext_resume_last:n {#1}
1639            }
1640        }
1641        {
1642          \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str _tl }
1643          \tl_gset:ce { g__enumext_series_ \l__enumext_series_str _tl }
1644            { \__enumext_filter_series:n {#1} }
1645          \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str _int }
1646            {
1647              \int_new:c { g__enumext_series_ \l__enumext_series_str _int }
1648            }
1649        }
1650    }
```

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.

```
1651 \cs_new_protected:Npn \__enumext_resume_last:n #1
1652    {
1653      \bool_if:NT \l__enumext_standar_first_bool
1654        {
1655          \tl_gclear:N \g__enumext_standar_series_tl
1656          \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1657        }
1658      \bool_if:NT \l__enumext_starred_first_bool
1659        {
1660          \tl_gclear:N \g__enumext_starred_series_tl
1661          \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1662        }
1663    }
```

(*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).

```
1664  \cs_new_protected:Nn \__enumext_resume_save_counter:
1665    {
1666      \bool_if:NT \g__enumext_standar_bool
1667        {
1668          \tl_if_empty:NF \l__enumext_series_str
1669            {
1670              \int_gset_eq:cN
1671                { g__enumext_series_ \l__enumext_series_str _int } \value{enumXi}
1672            }
1673          \tl_if_empty:NTF \l__enumext_resume_name_tl
1674            {
1675              \str_if_empty:NT \l__enumext_series_str
1676                {
1677                  \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1678                }
1679            }
1680            {
1681              \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1682                {
1683                  \int_gset_eq:cN
1684                    { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXi}
1685                }
1686            }
1687          \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1688            {
1689              \int_gset_eq:cN
1690                { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1691            }
1692        }
1693      \bool_if:NT \g__enumext_starred_bool
1694        {
1695          \tl_if_empty:NF \l__enumext_series_str
1696            {
1697              \int_gset_eq:cN
1698                { g__enumext_series_ \l__enumext_series_str _int } \value{enumXvii}
1699            }
1700          \tl_if_empty:NTF \l__enumext_resume_name_tl
1701            {
1702              \str_if_empty:NT \l__enumext_series_str
1703                {
1704                  \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1705                }
1706            }
1707            {
1708              \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1709                {
1710                  \int_gset_eq:cN
1711                    { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXvii}
1712                }
1713            }
1714          \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1715            {
1716              \int_gset_eq:cN
1717                { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1718            }
1719        }
1720    }
```

(*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.

```
1721 \cs_new_protected:Npn \__enumext_resume_series:n #1
1722   {
1723     \tl_if_empty:nTF {#1}
1724       {
1725         \__enumext_resume_counter:n { }
1726       }
1727       {
1728         \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1729           {
1730             \__enumext_resume_counter:n {#1}
1731             \bool_if:NT \g__enumext_standar_bool
1732               {
1733                 \keys_set:nv { enumext / level-1 }
1734                   { g__enumext_series_ \tl_to_str:n {#1} _tl }
1735               }
1736             \bool_if:NT \g__enumext_starred_bool
1737               {
1738                 \keys_set:nv { enumext / enumext* }
1739                   { g__enumext_series_ \tl_to_str:n {#1} _tl }
1740               }
1741           }
1742           {
1743             \bool_if:NT \g__enumext_standar_bool
1744               {
1745                 \msg_error:nnn { enumext } { unknown-series } {#1}
1746               }
1747             \bool_if:NT \g__enumext_starred_bool
1748               {
1749                 \msg_error:nnn { enumext } { unknown-series } {#1}
1750               }
1751           }
1752       }
1753   }
```

(*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.

```
1754 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1755   {
1756     \bool_set_true:N \l__enumext_resume_active_bool
1757     \tl_set:Nn \l__enumext_resume_name_tl {#1}
1758     \tl_if_empty:NTF \l__enumext_resume_name_tl
1759       {
1760         \__enumext_resume_counter:
1761       }
1762       {
1763         \__enumext_resume_counter_series:
1764       }
1765     \__enumext_resume_counter_save_ans:
1766   }
```

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.

```
1767 \cs_new_protected:Nn \__enumext_resume_counter:
1768   {
1769     \bool_if:NT \g__enumext_standar_bool
```

```
1770        {
1771          \int_gincr:N \g__enumext_resume_int
1772          \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1773        }
1774      \bool_if:NT \g__enumext_starred_bool
1775        {
1776          \int_gincr:N \g__enumext_resume_vii_int
1777          \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1778        }
1779    }
```

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.

```
1780  \cs_new_protected:Nn \__enumext_resume_counter_series:
1781    {
1782      \bool_if:NT \g__enumext_standar_bool
1783        {
1784          \int_set:Nn \l__enumext_start_i_int
1785            {
1786              \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1787            }
1788        }
1789      \bool_if:NT \g__enumext_starred_bool
1790        {
1791          \int_set:Nn \l__enumext_start_vii_int
1792            {
1793              \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1794            }
1795        }
1796    }
```

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.

```
1797  \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1798    {
1799      \bool_lazy_and:nnT
1800        { \bool_if_p:N \l__enumext_standar_first_bool }
1801        { \bool_if_p:N \l__enumext_store_active_bool }
1802        {
1803          \int_set:Nn \l__enumext_start_i_int
1804            {
1805              \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1806            }
1807        }
1808      \bool_lazy_and:nnT
1809        { \bool_if_p:N \l__enumext_starred_first_bool }
1810        { \bool_if_p:N \l__enumext_store_active_bool }
1811        {
1812          \int_set:Nn \l__enumext_start_vii_int
1813            {
1814              \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1815            }
1816        }
1817    }
```

(*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.

```
1818  \cs_new_protected:Nn \__enumext_resume_starred:
1819    {
1820      \bool_if:NT \g__enumext_standar_bool
1821        {
1822          \tl_if_empty:NF \g__enumext_standar_series_tl
1823            {
1824              \__enumext_resume_counter:n { }
```

```
1825        \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1826      }
1827    }
1828  \bool_if:NT \g__enumext_starred_bool
1829    {
1830      \tl_if_empty:NF \g__enumext_starred_series_tl
1831        {
1832          \__enumext_resume_counter:n { }
1833          \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1834        }
1835    }
1836  }
```

(*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*.

```
1837 \cs_set_protected:Npn \__enumext_tmp:n #1
1838   {
1839     \keys_define:nn { enumext / #1 }
1840       {
1841         save-ans .code:n = \__enumext_storing_set:n {##1},
1842         save-ans .value_required:n = true,
1843       }
1844   }
1845 \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 dis-
__enumext_stop_save_ans_msg: play 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:`.

```
1846 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
1847   {
1848     \msg_term:nnVV { enumext } { save-ans-log }
1849       \g__enumext_envir_name_tl \l__enumext_store_name_tl
1850   }
1851 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
1852   {
1853     \msg_term:nnVV { enumext } { save-ans-log-hook }
1854       \g__enumext_envir_name_tl \g__enumext_store_name_tl
1855   }
```

(*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
__enumext_storing_exec: `\l__enumext_store_name_tl` which will contain the *"store name"* of the ⟨*sequence*⟩ and ⟨*prop list*⟩ we will use. If `\l__enumext_store_name_tl` is *empty* we return an error message, otherwise will return the appropriate message `\__enumext_start_save_ans_msg:` and proceed to execute the function `\__enumext_storing_exec:` for enumext and enumext* environments.

```
1856 \cs_new_protected:Npn \__enumext_storing_set:n #1
1857   {
1858     \tl_set:Ne \l__enumext_store_name_tl {#1}
1859     \tl_if_empty:NTF \l__enumext_store_name_tl
1860       {
1861         \bool_lazy_or:nnT
1862           { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1863           {
1864             \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
1865           }
1866       }
1867       {
1868         \bool_lazy_or:nnT
```

```
1869              { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1870              {
1871                \__enumext_start_save_ans_msg:
1872                \__enumext_storing_exec:
1873              }
1874          }
1875      }
```

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*.

```
1876 \cs_new_protected:Nn \__enumext_storing_exec:
1877   {
1878     \bool_set_true:N \l__enumext_store_active_bool
1879     \bool_set_true:N \l__enumext_check_answers_bool
1880     \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1881     \__enumext_anskey_env_make:V \l__enumext_store_name_tl
1882     \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1883       {
1884         \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
1885         \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1886       }
1887     \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1888       {
1889         \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
1890         \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1891       }
1892     \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1893       {
1894         \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
1895         \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1896       }
1897   }
```

(*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.

```
1898 \cs_set_protected:Npn \__enumext_tmp:n #1
1899   {
1900     \keys_define:nn { enumext / #1 }
1901       {
1902         check-ans .bool_set:N = \l__enumext_check_ans_key_bool,
1903         check-ans .initial:n  = false,
1904         check-ans .value_required:n = true,
1905         no-store  .code:n = {
1906                             \bool_set_false:N \l__enumext_check_answers_bool
1907                             \bool_set_false:N \l__enumext_check_ans_key_bool
1908                           },
1909         no-store  .value_forbidden:n = true,
1910       }
1911   }
1912 \clist_map_inline:nn
1913   {
1914     level-1, level-2, level-3, level-4, enumext*
1915   }
1916   { \__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.

```
1917 \cs_new_protected:Nn \__enumext_check_ans_active:
1918   {
1919     \tl_if_empty:NF \l__enumext_store_name_tl
1920       {
1921         \bool_if:NT \l__enumext_check_answers_bool
1922           {
1923             \__enumext_check_ans_level:
1924           }
1925       }
1926   }
```

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"*.

```
1927 \cs_new_protected:Nn \__enumext_check_ans_level:
1928   {
1929     \int_case:nn { \l__enumext_level_int }
1930       {
1931         { 1 }{
1932             \bool_lazy_all:nT
1933               {
1934                 { \bool_if_p:N \g__enumext_starred_bool }
1935                 { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
1936               }
1937               {
1938                 \int_gdecr:N \g__enumext_item_number_int
1939                 \bool_set_false:N \l__enumext_item_number_bool
1940               }
1941           }
1942         { 2 }{
1943             \int_gdecr:N \g__enumext_item_number_int
1944             \bool_set_false:N \l__enumext_item_number_bool
1945           }
1946         { 3 }{
1947             \int_gdecr:N \g__enumext_item_number_int
1948             \bool_set_false:N \l__enumext_item_number_bool
1949           }
```

```
1950            { 4 }{
1951                    \int_gdecr:N \g__enumext_item_number_int
1952                    \bool_set_false:N \l__enumext_item_number_bool
1953                }
1954        }
```

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.

```
1955        \int_case:nn { \l__enumext_level_h_int }
1956        {
1957            { 1 }{
1958                    \bool_lazy_all:nT
1959                        {
1960                          { \bool_if_p:N \g__enumext_standar_bool }
1961                          { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
1962                        }
1963                        {
1964                          \int_gdecr:N \g__enumext_item_number_int
1965                          \bool_set_false:N \l__enumext_item_number_bool
1966                        }
1967                }
1968        }
1969    }
```

(*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.

```
1970  \cs_new_protected:Nn \__enumext_check_ans_key_hook:
1971    {
1972      \bool_lazy_and:nnT
1973        { \bool_if_p:N \l__enumext_check_ans_key_bool }
1974        { \bool_if_p:N \g__enumext_standar_bool }
1975        {
1976          \bool_gset_true:N \g__enumext_check_ans_key_bool
1977        }
1978      \bool_lazy_and:nnT
1979        { \bool_if_p:N \l__enumext_check_ans_key_bool }
1980        { \bool_if_p:N \g__enumext_starred_bool }
1981        {
1982          \bool_gset_true:N \g__enumext_check_ans_key_bool
1983        }
1984    }
```

(*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:.

```
1985  \cs_new_protected:Nn \__enumext_item_answer_diff:
1986    {
1987      \int_gset:Nn \g__enumext_item_answer_diff_int
1988        {
1989          \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
1990        }
1991    }
```

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

__enumext_check_ans_show:
__enumext_check_ans_msg_less:
__enumext_check_ans_msg_same_ok:
__enumext_check_ans_msg_greater:

The function __enumext_check_ans_show: will be executed within the function __enumext_-execute_after_env: when the key check-ans is active, that is, when \g__enumext_check_ans_-key_bool is *"true"* and will return the appropriate message according to the value of \g__enumext_-item_answer_diff_int set by the function __enumext_item_answer_diff:.

```
1992  \cs_new_protected:Nn \__enumext_check_ans_show:
1993    {
1994      \int_case:nn { \g__enumext_item_answer_diff_int }
1995        {
1996          { -1 }{ \__enumext_check_ans_msg_less:    }
1997          {  0 }{ \__enumext_check_ans_msg_same_ok: }
```

```
1998              {  1 }{ \__enumext_check_ans_msg_greater: }
1999            }
2000        }
2001  \cs_new_protected:Nn \__enumext_check_ans_msg_less:
2002        {
2003          \msg_warning:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2004            { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2005        }
2006  \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
2007        {
2008          \msg_term:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2009            { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2010        }
2011  \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
2012        {
2013          \msg_warning:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2014            { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2015        }
```

(*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:.

```
2016  \cs_new_protected:Nn \__enumext_check_ans_log:
2017        {
2018          \int_case:nn { \g__enumext_item_answer_diff_int }
2019            {
2020              { -1 }{ \__enumext_check_ans_log_msg_less:    }
2021              {  0 }{ \__enumext_check_ans_log_msg_same_ok: }
2022              {  1 }{ \__enumext_check_ans_log_msg_greater: }
2023            }
2024        }
2025  \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
2026        {
2027          \msg_log:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2028            { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2029        }
2030  \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
2031        {
2032          \msg_log:nneee { enumext } { items-same-answer }  { \g__enumext_store_name_tl }
2033            { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2034        }
2035  \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
2036        {
2037          \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2038            { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2039        }
```

(*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.

```
2040  \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
2041        {
2042          \int_compare:nNnT
2043            { \g__enumext_check_starred_cmd_int } = { 0 }
2044            {
2045              \msg_warning:nnnV
2046                { enumext } { missing-starred }{ #1 } \l__enumext_check_start_line_env_tl
2047            }
2048          \int_compare:nNnT
2049            { \g__enumext_check_starred_cmd_int } > { 1 }
2050            {
2051              \msg_warning:nnnV
2052                { enumext } { many-starred }{ #1 } \l__enumext_check_start_line_env_tl
2053            }
2054          \int_gzero:N \g__enumext_check_starred_cmd_int
```

```
2055        \tl_clear:N \l__enumext_check_start_line_env_tl
2056      }
```

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

12.26   Keys and functions associated with storage

<div style="float:left">wrap-ans
wrap-opt
save-sep
mark-ans
mark-pos
show-ans
mark-ref
save-ref</div>

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*.

```
2057 \cs_set_protected:Npn \__enumext_tmp:n #1
2058   {
2059     \keys_define:nn { enumext / #1 }
2060       {
2061         wrap-ans    .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
2062         wrap-ans    .initial:n =
2063                       {
2064                         \fbox{\parbox[t]{\dimeval{\itemwidth -2\fboxsep -2\fboxrule}}{##1}}
2065                       },
2066         wrap-ans    .value_required:n = true,
2067         wrap-opt    .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
2068         wrap-opt    .initial:n = [{##1}],
2069         wrap-opt    .value_required:n = true,
2070         save-sep    .tl_set:N  = \l__enumext_store_keyans_item_opt_sep_tl,
2071         save-sep    .initial:n = {, ~ },
2072         save-sep    .value_required:n = true,
2073         mark-ans    .tl_set:N  = \l__enumext_mark_answer_sym_tl,
2074         mark-ans    .initial:n = \textasteriskcentered,
2075         mark-ans    .value_required:n = true,
2076         mark-pos    .choice:,
2077         mark-pos / left    .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2078         mark-pos / right   .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2079         mark-pos / unknown .code:n =
2080                            \msg_error:nneee { enumext } { unknown-choice }
2081                              { mark-pos } { left, ~ right } { \exp_not:n {##1} },
2082         mark-pos    .initial:n = right,
2083         mark-pos    .value_required:n = true,
2084         show-ans    .bool_set:N = \l__enumext_show_answer_bool,
2085         show-ans    .initial:n  = false,
2086         show-ans    .value_required:n = true,
2087         show-pos    .bool_set:N = \l__enumext_show_position_bool,
2088         show-pos    .initial:n  = false,
2089         show-pos    .value_required:n = true,
2090         mark-ref    .tl_set:N   = \l__enumext_mark_ref_sym_tl,
2091         mark-ref    .initial:n  = \textasteriskcentered,
2092         mark-ref    .value_required:n = true,
2093         save-ref    .bool_set:N = \l__enumext_store_ref_key_bool,
2094         save-ref    .initial:n  = false,
2095         save-ref    .value_required:n = true,
2096       }
2097   }
2098 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

<div style="float:left">mark-pos
show-ans
show-pos</div>

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

```
2099 \cs_set_protected:Npn \__enumext_tmp:n #1
2100   {
2101     \keys_define:nn { enumext / #1 }
2102       {
2103         mark-pos .choice:,
2104         mark-pos / left  .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2105         mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2106         mark-pos .initial:n = right,
2107         mark-pos .value_required:n  = true,
2108         show-ans .bool_set:N = \l__enumext_show_answer_bool,
2109         show-ans .initial:n  = false,
2110         show-ans .value_required:n = true,
2111         show-pos .bool_set:N = \l__enumext_show_position_bool,
2112         show-pos .initial:n  = false,
2113         show-pos .value_required:n = true,
```

```
2114            }
2115       }
2116  \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*⟩.

```
2117  \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2118    {
2119      \bool_if:cF { l__enumext_store_save_key_ \__enumext_level: _bool }
2120        {
2121          \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2122          \tl_set:ce
2123            { l__enumext_store_save_key_ \__enumext_level: _tl }
2124            { \__enumext_filter_save_key:n {#1} }
2125        }
2126    }
2127  \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2128    {
2129      \bool_if:NF \l__enumext_store_save_key_vii_bool
2130        {
2131          \tl_clear:N \l__enumext_store_save_key_vii_tl
2132          \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2133        }
2134    }
```

(*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.

```
2135  \cs_set_protected:Npn \__enumext_tmp:n #1
2136    {
2137      \keys_define:nn { enumext / enumext* }
2138        {
2139          save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2140          save-key .value_required:n = true,
2141        }
2142      \keys_define:nn { enumext / #1 }
2143        {
2144          save-key .code:n = \__enumext_parse_save_key:n {##1},
2145          save-key .value_required:n = true,
2146        }
2147    }
2148  \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*.

```
2149  \cs_new_protected:Npn \__enumext_parse_save_key:n #1
2150    {
2151      \bool_set_true:c { l__enumext_store_save_key_ \__enumext_level: _bool }
```

```
2152       \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2153       \tl_set:ce
2154         { l__enumext_store_save_key_ \__enumext_level: _tl }
2155         { \__enumext_filter_save_key:n {#1} }
2156    }
2157 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2158    {
2159      \bool_set_true:N \l__enumext_store_save_key_vii_bool
2160      \tl_clear:N \l__enumext_store_save_key_vii_tl
2161      \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2162    }
```

(*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.

```
2163 \cs_new:Npn \__enumext_filter_save_key:n #1
2164    {
2165      \use:e
2166        {
2167          \keyval_parse:NNn
2168            \__enumext_filter_save_key_key:n
2169            \__enumext_filter_save_key_pair:nn {#1}
2170        }
2171    }
```

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.

```
2172 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2173    {
2174      \str_case:nnF {#1}
2175        {
2176          { resume } {} { resume* } {} { no-store } {} { base-fix } {}
2177        }
2178        { , { \exp_not:n {#1} } }
2179    }
```

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.

```
2180 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2181    {
2182      \str_case:nnF {#1}
2183        {
2184          { series   } {} { resume    } {} { save-ans } {} { save-ref   } {}
2185          { save-key } {} { check-ans } {} { show-ans } {} { show-pos   } {}
2186          { wrap-ans } {} { mark-ans  } {} { wrap-opt } {} { save-sep   } {}
2187          { mark-ref } {} { mini-env  } {} { mini-sep } {} { mini-right } {}
2188          { mini-right* } {}
2189        }
2190        { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2191    }
```

(*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.

```
2192 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2193    {
2194      \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2195        {
2196          \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2197        }
2198        { #1 }
```

```
2199    }
2200 \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.

```
2201 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2202    {
2203       \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
2204    }
2205 \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.

```
2206 \cs_new_protected:Nn \__enumext_store_level_open:
2207    {
2208       \bool_if:NT \l__enumext_check_answers_bool
2209          {
2210             \tl_if_empty:cTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2211                {
2212                   \__enumext_store_addto_seq:n
2213                      {
2214                         \item \begin{enumext}
2215                      }
2216                }
2217                {
2218                   \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2219                      {
2220                         \item \begin{enumext} [
2221                      }
2222                   \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2223                      {
2224                         ]
2225                      }
2226                   \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2227                }
2228          }
2229    }
2230 \cs_new_protected:Nn \__enumext_store_level_close:
2231    {
2232       \bool_if:NT \l__enumext_check_answers_bool
2233          {
2234             \__enumext_store_addto_seq:n { \end{enumext} }
2235          }
2236    }
```

(*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.

```
2237 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2238    {
2239       \bool_if:NT \l__enumext_check_answers_bool
2240          {
2241             \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2242                {
2243                   \__enumext_store_addto_seq:n
2244                      {
2245                         \item \begin{enumext*}
2246                      }
2247                }
2248                {
```

```
2249          \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2250            {
2251              \item \begin{enumext*}[
2252            }
2253          \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2254            {
2255              ]
2256            }
2257          \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2258        }
2259      }
2260    }
2261 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2262    {
2263      \bool_if:NT \l__enumext_check_answers_bool
2264        {
2265          \__enumext_store_addto_seq:n { \end{enumext*} }
2266        }
2267    }
```

(*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

```
2268 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2269    {
2270      \mode_leave_vertical:
2271      \skip_horizontal:n { -\dim_use:N #2 }
2272      \makebox[0pt][ r ]
2273        {
2274          \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2275            {
2276              \tl_use:N \l__enumext_mark_answer_sym_tl
2277            }
2278        }
2279      \skip_horizontal:n { \dim_use:N #2 }
2280    }
2281 \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.

```
2282 \cs_new_protected:Nn \__enumext_store_internal_ref:
2283    {
2284      \cs_set_protected:Npn \__enumext_tmp:n ##1
2285        {
2286          \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2287          \tl_reverse:c { l__enumext_label_copy_##1_tl }
2288          \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2289          \tl_reverse:c { l__enumext_label_copy_##1_tl }
2290        }
2291      \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2292      \cs_set:Npn \__enumext_tmp:n ##1
2293        { . \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.

```
2294      \bool_lazy_all:nT
2295        {
```

```
2296              { \bool_if_p:N \g__enumext_starred_bool }
2297              { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2298            }
2299            {
2300              \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2301                { \tl_use:N \l__enumext_label_copy_vii_tl }
2302            }
2303        \bool_lazy_all:nT
2304          {
2305              { \bool_not_p:n { \g__enumext_standar_bool } }
2306              { \bool_if_p:N \l__enumext_standar_bool }
2307              { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2308          }
2309          {
2310              \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2311                {
2312                  \tl_use:N \l__enumext_label_copy_vii_tl
2313                  \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2314                }
2315          }
```

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.

```
2316        \bool_lazy_all:nT
2317          {
2318              { \bool_if_p:N \g__enumext_standar_bool }
2319              { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2320              { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2321          }
2322          {
2323              \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2324                {
2325                  \tl_use:N \l__enumext_label_copy_i_tl
2326                  \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2327                }
2328          }
2329        \cs_set:Npn \__enumext_tmp:n ##1
2330          { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } . }
2331        \bool_lazy_all:nT
2332          {
2333              { \bool_if_p:N \g__enumext_standar_bool }
2334              { \bool_if_p:N \l__enumext_starred_bool }
2335              { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2336          }
2337          {
2338              \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2339                {
2340                  \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2341                  \tl_use:N \l__enumext_label_copy_vii_tl
2342                }
2343          }
```

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

```
2344        \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2345          {
2346              \l__enumext_store_name_tl \c_colon_str
2347              \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2348          }
```

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.

```
2349        \tl_put_right:Ne \l__enumext_write_aux_file_tl
2350          {
2351              \__enumext_newlabel:nn
2352                { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2353                { \l__enumext_newlabel_arg_two_tl }
2354          }
2355        \l__enumext_write_aux_file_tl
2356      }
```

(*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*⟩.

```
2357  \cs_new_protected:Npn \__enumext_store_anskey_code:n #1
2358    {
2359      \int_gincr:N \g__enumext_item_anskey_int
2360      \__enumext_store_addto_prop:n {#1}
2361      \bool_if:NT \l__enumext_store_ref_key_bool
2362        {
2363          \__enumext_store_internal_ref:
2364        }
2365      \__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`.

```
2366      \tl_clear:N \l__enumext_store_anskey_arg_tl
2367      \bool_lazy_and:nnT
2368        { \bool_if_p:N \l__enumext_store_columns_break_bool }
2369        { \bool_not_p:n { \l__enumext_starred_bool } }
2370        {
2371          \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2372        }
2373      \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`.

```
2374      \bool_lazy_and:nnT
2375        { \bool_not_p:n { \l__enumext_starred_bool } }
2376        { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2377        {
2378          \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2379            {
2380              ( \exp_not:V \l__enumext_store_item_join_int )
2381            }
2382        }
```

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*`.

```
2383      \bool_if:NTF \l__enumext_store_item_star_bool
2384        {
2385          \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2386          \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2387            {
2388              \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2389                {
2390                  [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2391                }
2392            }
2393          \dim_compare:nT
2394            {
2395              \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2396            }
2397            {
2398              \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2399                {
2400                  [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2401                }
2402            }
2403          \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2404        }
2405        {
2406          \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2407        }
```

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*⟩.

```
2408      \bool_lazy_and:nnT
```

```
2409          { \bool_if_p:N \l__enumext_store_ref_key_bool }
2410          { \bool_if_p:N \l__enumext_hyperref_bool }
2411          {
2412            \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2413              {
2414                \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2415                    { \exp_not:V \l__enumext_mark_ref_sym_tl }
2416              }
2417          }
2418      \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2419    }
```

(*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.

```
2420  \cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
2421    {
2422      \par
2423      \bool_if:NTF \l__enumext_starred_bool
2424        {
2425          \__enumext_print_keyans_box:NN \l__enumext_labelwidth_vii_dim  \l__enumext_labelsep_vii_d
2426        }
2427        {
2428          \__enumext_print_keyans_box:cc
2429            { l__enumext_labelwidth_ \__enumext_level: _dim }
2430            { l__enumext_labelsep_ \__enumext_level: _dim }
2431        }
2432      \__enumext_anskey_wrapper:n { #1 }
2433    }
```

(*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.

```
2434  \cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
2435    {
2436      \bool_if:NT \l__enumext_show_answer_bool
2437        {
2438          \__enumext_anskey_show_wrap_arg:n { #1 }
2439        }
2440      \bool_if:NT \l__enumext_show_position_bool
2441        {
2442          \tl_set:Ne \l__enumext_mark_answer_sym_tl
2443            {
2444              \group_begin:
2445              \exp_not:N \normalfont
2446              \exp_not:N \footnotesize [ \int_eval:n
2447                {
2448                  \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2449                }
2450              ]
2451              \group_end:
2452            }
2453          \__enumext_anskey_show_wrap_arg:n { #1 }
2454        }
2455    }
```

(*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`
`\__enumext_anskey_unknown:nn`

First we'll add the keys `break-col`, `item-join`, `item-star`, `item-sym*` and `item-pos*`.

```
2456 \keys_define:nn { enumext / anskey }
2457   {
2458     break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2459     break-col .default:n  = true,
2460     break-col .value_forbidden:n = true,
2461     item-join .int_set:N  = \l__enumext_store_item_join_int,
2462     item-join .value_required:n = true,
2463     item-star .bool_set:N = \l__enumext_store_item_star_bool,
2464     item-star .default:n  = true,
2465     item-star .value_forbidden:n = true,
2466     item-sym* .tl_set:N   = \l__enumext_store_item_symbol_tl,
2467     item-sym* .value_required:n = true,
2468     item-pos* .dim_set:N  = \l__enumext_store_item_symbol_sep_dim,
2469     item-pos* .value_required:n = true,
2470     unknown   .code:n     = { \__enumext_anskey_unknown:n {#1} },
2471   }
```

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`.

```
2472 \cs_new_protected:Npn \__enumext_anskey_unknown:n #1
2473   {
2474     \exp_args:NV \__enumext_anskey_unknown:nn \l_keys_key_str {#1}
2475   }
2476 \cs_new_protected:Npn \__enumext_anskey_unknown:nn #1 #2
2477   {
2478     \tl_if_blank:nTF {#2}
2479       {
2480         \msg_error:nnn { enumext } { anskey-cmd-key-unknown } {#1}
2481       }
2482       {
2483         \msg_error:nnnn { enumext } { anskey-cmd-key-value-unknown } {#1} {#2}
2484       }
2485   }
```

(*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`.

```
2486 \NewDocumentCommand \anskey { o +m }
2487   {
2488     \__enumext_anskey_safe_outer:
2489     \group_begin:
2490       \bool_if:NT \l__enumext_check_answers_bool
2491         {
2492           \tl_if_novalue:nF {#1}
2493             {
2494               \keys_set:nn { enumext / anskey } {#1}
2495             }
2496           \tl_if_blank:nTF {#2}
2497             {
2498               \msg_error:nn { enumext } { anskey-empty-arg }
2499             }
2500             {
2501               \__enumext_anskey_safe_inner:
2502               \__enumext_store_anskey_code:n {#2}
2503             }
2504         }
2505     \group_end:
2506   }
```

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

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.

```
2507  \cs_new_protected:Nn \__enumext_anskey_safe_outer:
2508    {
2509      \bool_if:NF \l__enumext_store_active_bool
2510        {
2511          \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
2512        }
2513      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2514        {
2515          \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
2516        }
2517      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2518        {
2519          \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans* }
2520        }
2521      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2522        {
2523          \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
2524        }
2525    }
```

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.

```
2526  \cs_new_protected:Nn \__enumext_anskey_safe_inner:
2527    {
2528      \int_incr:N \l__enumext_anskey_level_int
2529      \int_compare:nNnT { \l__enumext_anskey_level_int } > { 1 }
2530        {
2531          \msg_error:nn { enumext } { anskey-nested }
2532        }
2533      \bool_if:NF \l__enumext_item_number_bool
2534        {
2535          \msg_error:nn { enumext } { anskey-unnumber-item }
2536        }
2537      \mode_if_math:T
2538        {
2539          \msg_error:nne { enumext } { anskey-math-mode } { \c_backslash_str anskey }
2540        }
2541    }
```

(*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.

```
2542  \cs_new_protected:Nn \__enumext_undefine_anskey_env:
2543    {
2544      \cs_undefine:c { anskey* }
2545      \cs_undefine:c { endanskey* }
2546      \cs_undefine:c { __scontents_anskey*_env_begin: }
2547      \cs_undefine:c { __scontents_anskey*_env_end: }
2548    }
```

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

```
2549 \__enumext_before_env:nn { enumext }
2550   {
2551     \bool_lazy_and:nnT
2552       { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2553       { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2554       {
2555         \cs_if_free:cF { __scontents_anskey*_env_begin: }
2556           {
2557             \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2558           }
2559       }
2560   }
2561 \__enumext_before_env:nn { enumext* }
2562   {
2563     \bool_lazy_and:nnT
2564       { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2565       { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2566       {
2567         \cs_if_free:cF { __scontents_anskey*_env_begin: }
2568           {
2569             \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2570           }
2571       }
2572   }
```

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.

```
2573 \__enumext_before_env:nn { anskey* }
2574   {
2575     \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2576       {
2577         \msg_error:nnn { enumext } { anskey-env-wrong }{ keyans }
2578       }
2579     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2580       {
2581         \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
2582       }
2583     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2584       {
2585         \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
2586       }
2587     \bool_if:NF \l__enumext_item_number_bool
2588       {
2589         \msg_error:nn { enumext } { anskey-unnumber-item }
2590       }
2591     \mode_if_math:T
2592       {
2593         \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
2594       }
2595   }
```

(*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.

```
2596 \cs_new_protected:Npn \__enumext_anskey_env_make:n #1
2597   {
2598     \bool_if:NT \l__enumext_anskey_env_bool
2599       {
2600         \newenvsc{anskey*}[store-env=#1,print-env=false]
2601         \__enumext_anskey_env_exec:
2602       }
2603   }
2604 \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`.

```
2605  \cs_new_protected:Nn \__enumext_anskey_env_define_keys:
2606    {
2607      \keys_define:nn { scontents / scontents }
2608        {
2609          break-col .bool_gset:N = \g__enumext_store_columns_break_bool,
2610          break-col .default:n   = true,
2611          break-col .value_forbidden:n = true,
2612          item-join .int_gset:N  = \g__enumext_store_item_join_int,
2613          item-join .value_required:n = true,
2614          item-star .bool_gset:N = \g__enumext_store_item_star_bool,
2615          item-star .default:n   = true,
2616          item-star .value_forbidden:n = true,
2617          item-sym* .tl_gset:N   = \g__enumext_store_item_symbol_tl,
2618          item-sym* .value_required:n = true,
2619          item-pos* .dim_gset:N  = \g__enumext_store_item_symbol_sep_dim,
2620          item-pos* .value_required:n = true,
2621          print-env .undefine:,
2622          store-env .undefine:,
2623          write-out .undefine:,
2624          unknown   .code:n      = { \__enumext_anskey_env_unknown:n {##1} },
2625        }
2626    }
```

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`.

```
2627  \cs_new_protected:Npn \__enumext_anskey_env_unknown:n #1
2628    {
2629      \exp_args:NV \__enumext_anskey_env_unknown:nn \l_keys_key_str {#1}
2630    }
2631  \cs_new_protected:Npn \__enumext_anskey_env_unknown:nn #1#2
2632    {
2633      \tl_if_blank:nTF {#2}
2634        {
2635          \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
2636        }
2637        {
2638          \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
2639        }
2640    }
```

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`.

```
2641  \cs_new_protected:Nn \__enumext_anskey_env_reset_keys:
2642    {
2643      \keys_define:nn { scontents / scontents }
2644        {
2645          break-col .undefine:,
2646          item-join .undefine:,
2647          item-star .undefine:,
2648          item-sym* .undefine:,
2649          item-pos* .undefine:,
2650          write-out .code:n     = {
2651                                    \bool_set_false:N \l__scontents_storing_bool
2652                                    \bool_set_true:N  \l__scontents_writing_bool
2653                                    \tl_set:Nn \l__scontents_fname_out_tl {##1}
2654                                  },
2655          write-out .value_required:n = true,
2656          print-env .meta:nn    = { scontents } { print-env = ##1 },
2657          print-env .default:n = true,
2658          store-env .meta:nn    = { scontents } { store-env = ##1 },
2659          unknown   .code:n     = { \__scontents_parse_environment_keys:n {##1} },
2660        }
2661    }
```

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*.

```
2662  \cs_new_protected:Npn \__enumext_rescan_anskey_env:n #1
2663    {
2664      \group_begin:
2665        \int_set:Nn \tex_newlinechar:D { `\^^J }
2666        \__scontents_rescan_tokens:x
2667          {
2668            \endgroup % This assumes \catcode`\\=0... Things might go off otherwise.
2669            #1
2670          }
2671    }
```

(*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*⟩.

```
2672  \cs_new_protected:Nn \__enumext_anskey_env_exec:
2673    {
2674      \__enumext_before_env:nn { anskey* }
2675        {
2676          \__enumext_anskey_env_define_keys:
2677        }
```

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.

```
2678      \hook_if_empty:nF {env/anskey*/after}
2679        {
2680          \hook_gremove_code:nn {env/anskey*/after} { * }
2681        }
2682      \__enumext_after_env:nn { anskey* }
2683        {
2684          \__enumext_anskey_env_save_keys:
2685          \tl_clear:N \l__enumext_store_anskey_env_tl
2686          \tl_clear:N \l__enumext_store_anskey_opt_tl
2687          \bool_if:NT \l__enumext_check_answers_bool
2688            {
2689              \tl_gset:Ne \l__enumext_store_anskey_env_tl
2690                {
2691                  \seq_item:ce { g__scontents_name_ \l__enumext_store_name_tl _seq } { -1 }
2692                }
2693              \regex_match:nVTF
2694                { ^\s* \z | ^\s* \u{c__scontents_hidden_space_str} \z }
2695                \l__enumext_store_anskey_env_tl
2696                {
2697                  \msg_error:nn { enumext } { anskey-empty-arg }
2698                }
2699                {
2700                  \__enumext_anskey_env_store:
2701                }
2702            }
2703          \__enumext_anskey_env_clean_vars:
2704          \__enumext_anskey_env_reset_keys:
2705        }
2706    }
```

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

```
2707  \cs_new_protected:Nn \__enumext_anskey_env_save_keys:
2708    {
2709      \bool_lazy_and:nnT
2710        { \bool_if_p:N \g__enumext_store_columns_break_bool }
2711        { \bool_not_p:n { \l__enumext_starred_bool } }
2712        {
2713          \tl_put_left:Ne \l__enumext_store_anskey_opt_tl { ,break-col, }
2714        }
```

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

```
2715    \bool_lazy_and:nnT
2716      { \bool_not_p:n { \l__enumext_starred_bool } }
2717      { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }
2718      {
2719        \tl_put_left::Ne \l__enumext_store_anskey_opt_tl
2720          {
2721            ,item-join = \exp_not:V \g__enumext_store_item_join_int,
2722          }
2723      }
```

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

```
2724    \bool_if:NT \g__enumext_store_item_star_bool
2725      {
2726        \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2727          {
2728            ,item-star,
2729          }
2730        \tl_if_empty:NF \g__enumext_store_item_symbol_tl
2731          {
2732            \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2733              {
2734                ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,
2735              }
2736          }
2737        \dim_compare:nT
2738          {
2739            \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
2740          }
2741          {
2742            \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2743              {
2744                ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
2745              }
2746          }
2747      }
2748    }
```

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`.

```
2749  \cs_new_protected:Nn \__enumext_anskey_env_store:
2750    {
2751      \group_begin:
2752        \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
2753          {
2754            \exp_args:Ne
2755              \__enumext_store_anskey_code:n
2756                {
2757                  \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2758                }
2759          }
2760          {
2761            \keys_set_known:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
2762            \exp_args:Ne
2763              \__enumext_store_anskey_code:n
2764                {
2765                  \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2766                }
2767          }
2768      \group_end:
2769    }
```

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

```
2770  \cs_new_protected:Nn \__enumext_anskey_env_clean_vars:
2771    {
2772      \bool_gset_false:N \g__enumext_store_columns_break_bool
2773      \int_gzero:N        \g__enumext_store_item_join_int
2774      \bool_gset_false:N \g__enumext_store_item_star_bool
```

```
2775      \tl_gclear:N        \g__enumext_store_item_symbol_tl
2776      \dim_gzero:N        \g__enumext_store_item_symbol_sep_dim
2777    }
```

(*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.

```
2778  \cs_new_protected:Nn \__enumext_execute_after_env:
2779    {
2780      \int_compare:nNnT { \l__enumext_level_int } = { 0 }
2781        {
2782          \tl_if_empty:NF \g__enumext_store_name_tl
2783            {
2784              \__enumext_stop_save_ans_msg:
2785              \__enumext_item_answer_diff:
2786              \__enumext_log_global_vars:
2787              \__enumext_log_answer_vars:
2788              \bool_if:NTF \g__enumext_check_ans_key_bool
2789                {
2790                  \__enumext_check_ans_show:
2791                }
2792                { \__enumext_check_ans_log: }
2793              \__enumext_undefine_anskey_env:
2794            }
2795          \__enumext_reset_global_vars:
2796        }
2797    }
```

(*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.

```
2798  \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2799    {
2800      \tl_clear:N \l__enumext_store_current_label_tl
2801      \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2802        {
2803          \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
2804        }
2805        {
2806          \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
2807        }
2808      \tl_if_novalue:nF { #1 }
2809        {
2810          % Set save-sep
2811          \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2812            {
2813              \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_op
2814            }
2815          \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2816        }
2817      \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
```

₂₈₁₈   }

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

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

The *"internal label and ref"* system for the keyans, keyans* and keyanspic environments has slight differences with the one implemented for the \anskey command, basically because in this environments we are interested in the current ⟨*label*⟩. The mechanism defined here will allow to execute \ref{⟨*store name : position*⟩} and will return 1.(A).

__enumext_keyans_store_ref:
__enumext_keyans_store_ref_aux_i:
__enumext_keyans_store_ref_aux_ii:

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

```
2819 \cs_new_protected:Nn \__enumext_keyans_store_ref:
2820   {
2821     \bool_if:NT \l__enumext_store_ref_key_bool
2822       {
2823         \cs_set_protected:Npn \__enumext_tmp:n ##1
2824           {
2825             \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2826             \tl_reverse:c { l__enumext_label_copy_##1_tl }
2827             \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2828             \tl_reverse:c { l__enumext_label_copy_##1_tl }
2829           }
2830         \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2831         \__enumext_keyans_store_ref_aux_i:
2832       }
2833   }
```

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.

```
2834 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2835   {
2836     \bool_if:NT \g__enumext_starred_bool
2837       {
2838         \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2839       }
2840     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2841       {
2842         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2843           { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
2844       }
2845     \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2846       {
2847         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2848           { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2849       }
2850     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2851       {
2852         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2853           { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2854       }
2855     \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2856       {
2857         \l__enumext_store_name_tl \c_colon_str
2858         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2859       }
2860     \__enumext_keyans_store_ref_aux_ii:
2861   }
```

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.

```
2862 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2863   {
2864     \tl_put_right:Ne \l__enumext_write_aux_file_tl
2865       {
2866         \__enumext_newlabel:nn
2867           { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2868           { \l__enumext_newlabel_arg_two_tl }
2869       }
```

```
2870          \l__enumext_write_aux_file_tl
2871       }
```

(*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.

```
2872  \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2873    {
2874       \tl_clear:N \l__enumext_store_current_label_tl
2875       \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2876         {
2877            \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
2878         }
2879         {
2880            \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
2881         }
2882       \tl_if_novalue:nF { #1 }
2883         {
2884            \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2885              {
2886                 \tl_put_right:Ne \l__enumext_store_current_label_tl
2887                   {
2888                      \l__enumext_store_keyans_item_opt_sep_tl
2889                   }
2890              }
2891            \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2892         }
2893       \__enumext_keyans_addto_seq_link:
2894    }
```

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.

```
2895  \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2896    {
2897       \bool_lazy_and:nnT
2898         { \bool_if_p:N \l__enumext_store_ref_key_bool }
2899         { \bool_if_p:N \l__enumext_hyperref_bool }
2900         {
2901            \tl_put_right:Ne \l__enumext_store_current_label_tl
2902              {
2903                 \hfill \exp_not:N \hyperlink
2904                   {
2905                      \exp_not:V \l__enumext_newlabel_arg_one_tl
2906                   }
2907                   { \exp_not:V \l__enumext_mark_ref_sym_tl }
2908              }
2909         }
2910       \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
2911       \bool_if:NT \l__enumext_check_answers_bool
2912         {
2913            \int_gincr:N \g__enumext_item_anskey_int
2914         }
2915    }
```

(*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.

```
2916  \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
2917    {
2918      \tl_if_novalue:nF { #1 }
2919        {
2920          \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
2921        }
2922      \bool_if:NT \l__enumext_show_answer_bool
2923        {
2924          \__enumext_keyans_show_ans:
2925        }
2926      \bool_if:NT \l__enumext_show_position_bool
2927        {
2928          \__enumext_keyans_show_pos:
2929        }
2930    }
2931  \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
2932    {
2933      \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
2934        {
2935          \bool_lazy_or:nnT
2936            { \bool_if_p:N \l__enumext_show_answer_bool }
2937            { \bool_if_p:N \l__enumext_show_position_bool }
2938            {
2939              \__enumext_keyans_wrapper_opt:n { \l__enumext_store_current_opt_arg_tl } \c_space_tl
2940            }
2941        }
2942    }
2943  \cs_new_protected:Nn \__enumext_keyans_show_ans:
2944    {
2945      \bool_if:NT \l__enumext_starred_bool
2946        {
2947          \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
2948          \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
2949        }
2950      \tl_put_left:Nn \l__enumext_label_v_tl
2951        {
2952          \__enumext_print_keyans_box:NN
2953            \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2954        }
2955    }
2956  \cs_new_protected:Nn \__enumext_keyans_show_pos:
2957    {
2958      \bool_if:NT \l__enumext_starred_bool
2959        {
2960          \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
2961          \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
2962        }
2963      \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2964        {
2965          \tl_set:Ne \l__enumext_mark_answer_sym_tl
2966            {
2967              \group_begin:
2968              \exp_not:N \normalfont
2969              \exp_not:N \footnotesize [ \int_eval:n
2970                {
2971                  \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2972                }
2973              ]
2974              \group_end:
2975            }
2976        }
2977        {
2978          \tl_set:Ne \l__enumext_mark_answer_sym_tl
2979            {
2980              \group_begin:
2981              \exp_not:N \normalfont
2982              \exp_not:N \footnotesize [ \int_eval:n
2983                {
2984                  \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
```

```
2985                    }
2986                  ]
2987               \group_end:
2988             }
2989         }
2990       \tl_put_left:Nn \l__enumext_label_v_tl
2991         {
2992           \__enumext_print_keyans_box:NN
2993             \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2994         }
2995     }
```

(*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.

```
2996 \cs_new_protected:Npn \__enumext_default_item:n #1
2997   {
2998     \tl_if_novalue:nTF {#1}
2999       {
3000         \bool_if:NT \l__enumext_check_answers_bool
3001           {
3002             \int_gincr:N \g__enumext_item_number_int
3003             \bool_set_true:N \l__enumext_item_number_bool
3004           }
3005         \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
3006         \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3007       }
3008       {
3009         \bool_set_eq:cc
3010           { l__enumext_wrap_label_ \__enumext_level: _bool }
3011           { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
3012         \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl
3013       }
3014   }
```

(*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.

```
3015 \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
3016   {
3017     \tl_if_novalue:nTF {#1}
3018       {
3019         \tl_gset_eq:Nc
3020           \g__enumext_item_symbol_aux_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
3021       }
3022       {
3023         \tl_gset:Nn \g__enumext_item_symbol_aux_tl {#1}
```

```
3024          }
3025        \tl_if_novalue:nTF {#2}
3026          {
3027            \dim_set_eq:cc
3028              { l__enumext_item_symbol_sep_ \__enumext_level: _dim }
3029              { l__enumext_labelsep_ \__enumext_level: _dim }
3030          }
3031          {
3032            \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
3033          }
3034        \bool_if:NT \l__enumext_check_answers_bool
3035          {
3036            \int_gincr:N \g__enumext_item_number_int
3037            \bool_set_true:N \l__enumext_item_number_bool
3038          }
3039        \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
3040        \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3041      }
```

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

```
3042  \cs_new_protected:Nn \__enumext_item_star_exec:
3043    {
3044      \tl_if_empty:cF { l__enumext_item_symbol_ \__enumext_level: _tl }
3045        {
3046          \mode_leave_vertical:
3047          \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3048          \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_tl }
3049          \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3050        }
3051    }
```

(*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*.

```
3052  \cs_new_protected:Nn \__enumext_redefine_item:
3053    {
3054      \RenewDocumentCommand \item { s o o }
3055        {
3056          \bool_if:nTF {##1}
3057            {
3058              \__enumext_starred_item:nn {##2} {##3}
3059            }
3060            { \__enumext_default_item:n {##2} }
3061        }
3062    }
```

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

```
3063  \cs_new_protected:Nn \__enumext_make_label:
3064    {
3065      \RenewDocumentCommand \makelabel { m }
3066        {
3067          \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
3068          \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3069          \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3070            {
3071              \__enumext_item_star_exec:
3072              \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3073            }
3074            { ##1 }
3075          \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
3076          \tl_gclear:N \g__enumext_item_symbol_aux_tl
3077        }
3078    }
```

(*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*.

```
3079 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
3080   {
3081     \keys_define:nn { enumext / #1 }
3082       {
3083         item-sym* .tl_set:c  = { l__enumext_item_symbol_#2_tl },
3084         item-sym* .value_required:n = true,
3085         item-sym* .initial:n  = {$\star$},
3086         item-pos* .dim_set:c  = { l__enumext_item_symbol_sep_#2_dim },
3087         item-pos* .value_required:n = true,
3088       }
3089   }
3090 \clist_map_inline:nn
3091   {
3092     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
3093   }
3094   { \__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.

```
3095 \cs_set_protected:Npn \__enumext_tmp:n #1
3096   {
3097     \keys_define:nn { enumext / #1 }
3098       {
3099         unknown .code:n = { \__enumext_keyans_unknown_keys:n {##1} }
3100       }
3101   }
3102 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }
```

Internal functions for handling unknown key.

```
3103 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:n #1
3104   {
3105     \exp_args:NV \__enumext_keyans_unknown_keys:nn \l_keys_key_str {#1}
3106   }
3107 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:nn #1#2
3108   {
3109     \tl_if_blank:nTF {#2}
3110       {
3111         \msg_error:nnn { enumext } { keyans-unknown-key } {#1}
3112       }
3113       {
3114         \msg_error:nnnn { enumext } { keyans-unknown-key-value } {#1} {#2}
3115       }
3116   }
```

(*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.

```
3117 \keys_define:nn { enumext / enumext* }
3118   {
3119     unknown .code:n = { \__enumext_starred_unknown_keys:n {#1} }
3120   }
```

Internal functions for handling unknown key.

```
3121 \cs_new_protected:Npn \__enumext_starred_unknown_keys:n #1
3122   {
3123     \exp_args:NV \__enumext_starred_unknown_keys:nn \l_keys_key_str {#1}
3124   }
3125 \cs_new_protected:Npn \__enumext_starred_unknown_keys:nn #1#2
```

```
3126    {
3127      \tl_if_blank:nTF {#2}
3128        {
3129          \msg_error:nnn { enumext } { starred-unknown-key } {#1}
3130        }
3131        {
3132          \msg_error:nnnn { enumext } { starred-unknown-key-value } {#1} {#2}
3133        }
3134    }
```

(*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.

```
3135  \cs_set_protected:Npn \__enumext_tmp:n #1
3136    {
3137      \keys_define:nn { enumext / #1 }
3138        {
3139          unknown .code:n = { \__enumext_standar_unknown_keys:n {##1} }
3140        }
3141    }
3142  \clist_map_inline:nn { level-1,level-2,level-3,level-4 } { \__enumext_tmp:n {#1} }
```

Internal functions for handling unknown key.

```
3143  \cs_new_protected:Npn \__enumext_standar_unknown_keys:n #1
3144    {
3145      \exp_args:NV \__enumext_standar_unknown_keys:nn \l_keys_key_str {#1}
3146    }
3147  \cs_new_protected:Npn \__enumext_standar_unknown_keys:nn #1#2
3148    {
3149      \tl_if_blank:nTF {#2}
3150        {
3151          \msg_error:nnn { enumext } { standar-unknown-key } {#1}
3152        }
3153        {
3154          \msg_error:nnnn { enumext } { standar-unknown-key-value } {#1} {#2}
3155        }
3156    }
```

(*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.

```
3157  \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
3158    {
3159      \tl_if_novalue:nTF { #1 }
3160        {
3161          \bool_set_true:N \l__enumext_wrap_label_v_bool
3162          \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
3163        }
3164        {
3165          \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
3166          \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
3167        }
3168    }
```

(*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*⟩.

```
3169  \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
3170    {
3171      \tl_set_eq:NN \l__enumext_store_current_label_tmp_tl \l__enumext_label_v_tl
3172      \__enumext_keyans_show_left:n { #1 }
3173      \bool_set_true:N \l__enumext_wrap_label_v_bool
3174      \__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*⟩.

```
3175    \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_store_current_label_tmp_tl
3176    \__enumext_keyans_addto_prop:n { #1 }
3177    \__enumext_keyans_store_ref:
3178    \__enumext_keyans_addto_seq:n { #1 }
3179    \int_gincr:N \g__enumext_check_starred_cmd_int
3180  }
```

(*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.

```
3181  \cs_new_protected:Nn \__enumext_keyans_redefine_item:
3182    {
3183      \RenewDocumentCommand \item { s o }
3184        {
3185          \bool_if:nTF {##1}
3186            {
3187              \peek_remove_spaces:n
3188                {
3189                  \__enumext_keyans_starred_item:n {##2}
3190                }
3191            }
3192            {
3193              \__enumext_keyans_default_item:n {##2}
3194            }
3195        }
3196    }
```

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

```
3197  \cs_new_protected:Nn \__enumext_keyans_make_label:
3198    {
3199      \RenewDocumentCommand \makelabel { m }
3200        {
3201          \tl_use:N \l__enumext_label_fill_left_v_tl
3202          \tl_use:N \l__enumext_label_font_style_v_tl
3203          \bool_if:NTF \l__enumext_wrap_label_v_bool
3204            {
3205              \__enumext_wrapper_label_v:n { ##1 }
3206            }
3207            { ##1 }
3208          \tl_use:N \l__enumext_label_fill_right_v_tl
3209        }
3210    }
```

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

🌶 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.
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.
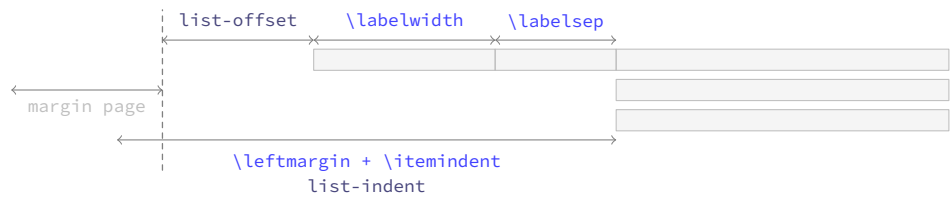
Figure 9: Representation of standard horizontal lengths in list environment.
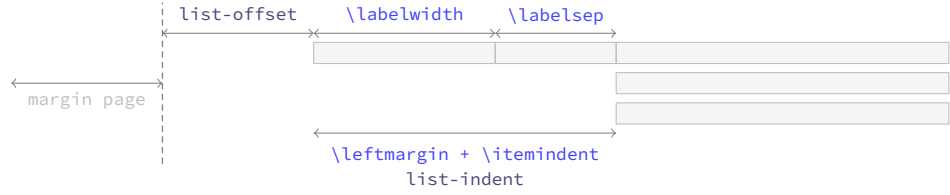
Figure 10: Representation of horizontal lengths concept in list in enumext.

__enumext_calc_hspace:NNNNNNN

__enumext_calc_hspace:ccccccc

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

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

And returns the *"adjusted"* values of \leftmargin and \itemindent.

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).

```
3211  \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
3212    {
3213      \dim_compare:nNnT { #1 } < { \c_zero_dim }
3214        {
3215          \msg_warning:nnnV { enumext } { width-non-positive }{ labelwidth }{ #1 }
3216          \dim_set:Nn #1 { \dim_abs:n { #1 } }
3217        }
3218      \dim_compare:nNnT { #2 } < { \c_zero_dim }
3219        {
3220          \msg_warning:nnnV { enumext } { width-negative }{ labelsep }{ #2 }
3221          \dim_set:Nn #2 { \dim_abs:n { #2 } }
3222        }
```

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

```
3223      \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2} }
```

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

```
3224      \dim_compare:nNnTF { #4 } < { \c_zero_dim }
3225        {
3226          \dim_set:Nn #6 { #1 + #2 - #4}
3227          \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3228        }
3229        {
3230          \dim_compare:nNnT { #4 } = { #1 + #2 }
3231            { \dim_set:Nn #6 { \c_zero_dim } }
3232          \dim_compare:nNnT { #4 } < { #1 + #2 }
3233            { \dim_set:Nn #6 { #1 + #2 - #4} }
3234          \dim_compare:nNnT { #4 } > { #1 + #2 }
3235            {
```
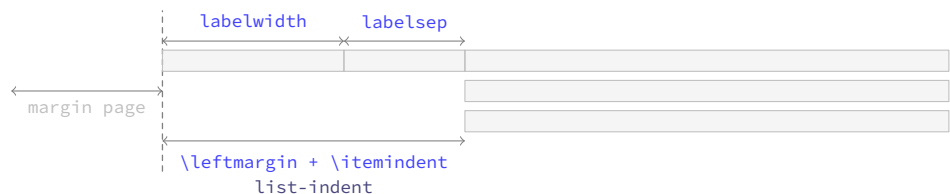

Figure 11: Default horizontal lengths in enumext.

```
3236          \dim_set:Nn #6 { -#1 - #2 + #4}
3237          \dim_set:Nn #6 { #6*-1}
3238        }
3239      \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3240    }
3241  }
3242 \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.

```
3243 \cs_set_protected:Npn \__enumext_tmp:n #1
3244   {
3245     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3246       {
3247         \__enumext_calc_hspace:ccccccc
3248           { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3249           { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3250           { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3251           { l__enumext_leftmargin_tmp_#1_bool }
3252         \clist_map_inline:nn
3253           { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3254           { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
3255         \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3256           { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
3257         \usecounter { enumX#1 }
3258         \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
3259         \str_if_eq:nnTF {#1} { v }
3260           {
3261             \__enumext_keyans_redefine_item:
3262             \__enumext_keyans_make_label:
3263             \__enumext_keyans_ref:
3264             \__enumext_keyans_fake_item:
3265             \bool_if:cT { l__enumext_show_length_#1_bool }
3266               {
3267                 \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3268               }
3269           }
3270           {
3271             \__enumext_redefine_item:
3272             \__enumext_make_label:
3273             \__enumext_standar_ref:
3274             \__enumext_fake_item:
3275             \bool_if:cT { l__enumext_show_length_#1_bool }
3276               {
3277                 \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \l__enumext_level_i
3278               }
3279           }
3280       }
3281   }
3282 \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.

```
3283 \cs_set_protected:Npn \__enumext_tmp:n #1
3284   {
3285     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3286       {
3287         \bool_set_true:c { l__enumext_leftmargin_tmp_#1_bool }
3288         \dim_zero:c { l__enumext_leftmargin_tmp_#1_dim }
3289         \__enumext_calc_hspace:ccccccc
3290           { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3291           { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3292           { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3293           { l__enumext_leftmargin_tmp_#1_bool }
```

```
3294        \clist_map_inline:nn
3295          { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3296          { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
3297        \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3298          { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
3299        \skip_set_eq:Nc \parsep  { l__enumext_itemsep_#1_skip }
3300        \skip_zero:N \partopsep
3301        \usecounter { enumX#1 }
3302        \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
3303        \__enumext_starred_ref:
3304        \str_if_eq:nnTF {#1} { vii }
3305          {
3306            \__enumext_fake_item_vii:
3307            \bool_if:cT { l__enumext_show_length_vii_bool }
3308              { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3309          }
3310          {
3311            \__enumext_fake_item_viii:
3312            \bool_if:cT { l__enumext_show_length_#1_bool }
3313              { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3314          }
3315      }
3316    }
3317  \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.

```
3318  \NewDocumentEnvironment{enumext}{ O{} }
3319    {
3320      \__enumext_safe_exec:
3321      \__enumext_parse_keys:n {#1}
3322      \__enumext_before_list:
3323      \__enumext_start_store_level:
3324      \__enumext_start_list:nn
3325        { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
3326        {
3327          \use:c { __enumext_list_arg_two_ \__enumext_level: : }
3328          \__enumext_before_keys_exec:
3329        }
3330      \__enumext_set_item_width:
3331      \__enumext_after_args_exec:
3332    }
3333    {
3334      \__enumext_stop_list:
3335      \__enumext_stop_store_level:
3336      \__enumext_after_list:
3337    }
```

(*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.

```
3338  \cs_new_protected:Nn \__enumext_set_item_width:
3339    {
3340      \dim_set:Nn \itemwidth
3341        {
3342          \linewidth
3343        }
3344      \dim_compare:nT
3345        {
3346          \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim } != \c_zero_dim
3347        }
3348        {
3349          \dim_sub:Nn \itemwidth
3350            {
3351              \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3352            }
3353        }
3354    }
```

(*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_env*`, 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"*.

```
3355  \cs_new_protected:Nn \__enumext_safe_exec:
3356    {
3357      \__enumext_internal_mini_page:
3358      \__enumext_is_not_nested:
3359      \int_incr:N \l__enumext_level_int
3360      \int_compare:nNnT { \l__enumext_level_int } > { 4 }
3361        { \msg_fatal:nn { enumext } { list-too-deep } }
3362      \bool_set_true:N \l__enumext_standar_bool
3363      \bool_set_false:N \l__enumext_starred_bool
3364      \__enumext_is_on_first_level:
3365    }
```

(*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.

```
3366  \cs_new_protected:Npn \__enumext_parse_keys:n #1
3367    {
3368      \tl_if_novalue:nF {#1}
3369        {
3370          \str_clear:N \l__enumext_series_str
3371          \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
3372            {
3373              \keys_set:nn { enumext / level-1 } {#1}
3374              \__enumext_parse_series:n {#1}
3375              \__enumext_nested_base_line_fix:
3376            }
3377            {
3378              \exp_args:Ne \keys_set:nn
3379                { enumext / level-\int_use:N \l__enumext_level_int } {#1}
3380            }
3381          \__enumext_store_active_keys:n {#1}
3382        }
3383    }
```

(*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*`.

```
3384  \cs_new_protected:Nn \__enumext_start_store_level:
3385    {
3386      \bool_lazy_all:nT
3387        {
3388          { \bool_if_p:N \l__enumext_store_active_bool }
3389          { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3390          { \bool_if_p:N \g__enumext_standar_bool }
3391        }
3392        {
3393          \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3394            {
3395              \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3396              \__enumext_store_level_open:
3397            }
3398        }
```

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

```
3399       \bool_lazy_all:nT
3400         {
3401           { \bool_if_p:N \l__enumext_store_active_bool }
3402           { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3403           { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
3404         }
3405         {
3406           \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3407             {
3408               \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3409               \__enumext_store_level_open:
3410             }
3411         }
3412     }
```

Close the stored structure.

```
3413 \cs_new_protected:Nn \__enumext_stop_store_level:
3414   {
3415     \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
3416       {
3417         \__enumext_store_level_close:
3418       }
3419   }
```

(*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.

```
3420 \cs_new_protected:Nn \__enumext_before_list:
3421   {
3422     \__enumext_vspace_above:
3423     \__enumext_before_args_exec:
3424     \__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_env* 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_env* environment on the *"left side"*, always having a current \linewidth as *maximum width* between them.

```
3425       \dim_compare:nNnT
3426         { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3427         {
3428           \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
3429             {
3430               \linewidth
3431               - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
3432               - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
3433             }
```

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_mini_addvspace: is called and the __enumext_mini_env* 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.

💣 Here we use the plain TEX macro \nointerlineskip to prevent baseline *"glue"* being added between the next pair of boxes in a *vertical list*.

```
3434           \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
3435           \int_gincr:N \g__enumext_minipage_stat_int
3436           \__enumext_mini_addvspace:
3437           \nointerlineskip\noindent
3438           \begin{__enumext_mini_env*}
3439             { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
3440         }
3441     \__enumext_multicols_start:
3442   }
```

(*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.

```
3443  \cs_new_protected:Nn \__enumext_multicols_start:
3444    {
3445      \int_compare:nNnT
3446        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3447        {
3448          \dim_compare:nNnT
3449            { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3450            {
3451              \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
3452                {
3453                  ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
3454                    + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
3455                  ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
3456                  - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3457                }
3458            }
3459          \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim  }
3460          \skip_zero:N \multicolsep
3461          \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3462            {
3463              \dim_zero:N \columnseprule
3464            }
```

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.

```
3465          \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3466            {
3467              \__enumext_multi_addvspace:
3468            }
3469          \raggedcolumns
3470          \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
3471        }
3472    }
```

(*End of definition for* __enumext_multicols_start:.)

__enumext_multicols_stop:

The function __enumext_multicols_stop: will stop the multicols environment. If the boolean variable \l__enumext_minipage_active_X_bool is false (not nested in __enumext_mini_env*) we will apply our *"vertical adjust"* spacing.

```
3473  \cs_new_protected:Nn \__enumext_multicols_stop:
3474    {
3475      \int_compare:nNnT
3476        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3477        {
3478          \end{multicols}
3479          \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3480            {
3481              \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
3482            }
3483        }
3484    }
```

(*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_env* environment has not been closed), then close __enumext_mini_env* and add the *adjusted vertical space* \l__enumext_minipage_after_skip, otherwise we will close the multicols environment.

```
3485  \cs_new_protected:Nn \__enumext_after_list:
3486    {
3487      \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
3488        {
```

```
3489        \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3490          {
3491            \msg_warning:nn { enumext } { missing-miniright }
3492            \miniright
3493          }
3494        \int_gzero:N \g__enumext_minipage_stat_int
3495        \end{__enumext_mini_env*}
3496        \par\addvspace { \l__enumext_minipage_after_skip }
3497      }
3498    { \__enumext_multicols_stop: }
```

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.

```
3499      \__enumext_after_stop_list:
3500      \__enumext_check_ans_key_hook:
3501      \__enumext_vspace_below:
3502      \bool_set_false:N \l__enumext_standar_bool
3503      \__enumext_resume_save_counter:
3504    }
```

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.

```
3505 \__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.

```
3506 \NewDocumentEnvironment{keyans}{ O{} }
3507    {
3508      \__enumext_keyans_safe_exec:
3509      \__enumext_keyans_parse_keys:n {#1}
3510      \__enumext_before_list_v:
3511      \__enumext_start_list:nn
3512        { \tl_use:N \l__enumext_label_v_tl }
3513        {
3514          \__enumext_list_arg_two_v:
3515          \__enumext_before_keys_exec_v:
3516        }
3517      \__enumext_keyans_set_item_width:
3518      \__enumext_after_args_exec_v:
3519    }
3520    {
3521      \__enumext_check_starred_cmd:n { item }
3522      \__enumext_stop_list:
3523      \__enumext_after_list_v:
3524    }
```

(*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.

```
3525 \cs_new_protected:Nn \__enumext_keyans_set_item_width:
3526    {
3527      \dim_set:Nn \itemwidth
3528        {
3529          \linewidth
3530        }
3531      \dim_compare:nT
3532        {
3533          \l__enumext_listoffset_v_dim != \c_zero_dim
3534        }
3535        {
3536          \dim_sub:Nn \itemwidth
3537            {
```

```
3538                        \l__enumext_listoffset_v_dim
3539                    }
3540                }
3541            }
```

(*End of definition for* __enumext_keyans_set_item_width:.)

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

```
3542 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3543    {
3544        \bool_if:NF \l__enumext_store_active_bool
3545            {
3546                \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
3547            }
3548        \int_incr:N \l__enumext_keyans_level_int
3549        \bool_set_true:N \l__enumext_keyans_env_bool
3550        \__enumext_keyans_name_and_start:
3551        % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3552        \bool_set_false:N \l__enumext_store_active_bool
3553        \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
3554            {
3555                \msg_error:nn { enumext } { keyans-nested }
3556            }
3557        \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3558            {
3559                \msg_error:nn { enumext } { keyans-wrong-level }
3560            }
3561    }
```

(*End of definition for* __enumext_keyans_safe_exec:.)

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

```
3562 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3563    {
3564        \keys_set:nn { enumext / keyans } {#1}
3565    }
```

(*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:

```
3566 \cs_new_protected:Nn \__enumext_before_list_v:
3567    {
3568        \__enumext_vspace_above_v:
3569        \__enumext_before_args_exec_v:
3570        \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3571            {
3572                \dim_set:Nn \l__enumext_minipage_left_v_dim
3573                    {
3574                        \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3575                    }
3576                \bool_set_true:N \l__enumext_minipage_active_v_bool
3577                \int_gincr:N \g__enumext_minipage_stat_int
3578                \__enumext_keyans_mini_addvspace:
3579                \nointerlineskip\noindent
3580                \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
3581            }
3582        \__enumext_keyans_multicols_start:
3583    }
3584 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3585    {
3586        \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3587            {
3588                \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3589                    {
3590                        \dim_set:Nn \l__enumext_columns_sep_v_dim
3591                            {
3592                                (
3593                                    \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3594                                ) / \l__enumext_columns_v_int
```

```
3595                - \l__enumext_listoffset_v_dim
3596              }
3597            }
3598          \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim
3599          \skip_zero:N \multicolsep
3600          \dim_zero:N \columnseprule % no rule here
3601          \bool_if:NF \l__enumext_minipage_active_v_bool
3602            {
3603              \__enumext_keyans_multi_addvspace:
3604            }
3605          \raggedcolumns
3606          \begin{multicols}{ \l__enumext_columns_v_int }
3607        }
3608    }
3609 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3610    {
3611      \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3612        {
3613          \end{multicols}
3614          \bool_if:NF \l__enumext_minipage_active_v_bool
3615            {
3616              \par\addvspace{ \l__enumext_multicols_below_v_skip }
3617            }
3618        }
3619    }
3620 \cs_new_protected:Nn \__enumext_after_list_v:
3621    {
3622      \bool_if:NTF \l__enumext_minipage_active_v_bool
3623        {
3624          \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3625            {
3626              \msg_warning:nn { enumext } { missing-miniright }
3627              \miniright
3628            }
3629          \int_gzero:N \g__enumext_minipage_stat_int
3630          \end{__enumext_mini_env*}
3631          \par\addvspace{ \l__enumext_minipage_after_skip }
3632        }
3633        {
3634          \__enumext_keyans_multicols_stop:
3635        }
3636      \bool_set_false:N \l__enumext_keyans_env_bool
3637      \__enumext_after_stop_list_v:
3638      \__enumext_vspace_below_v:
3639    }
```

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

12.40   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.
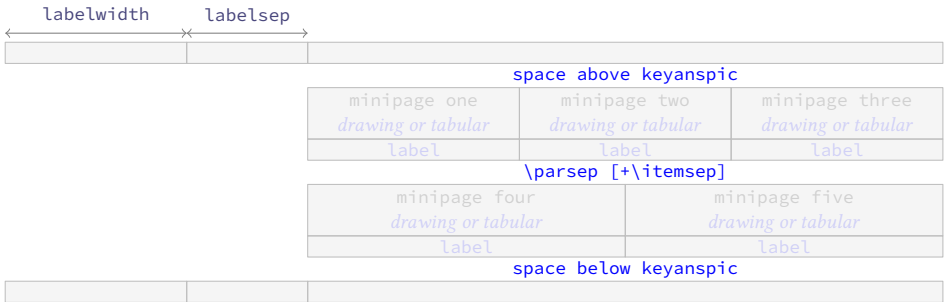
Figure 12: Representation of the `keyanspic` spacing in enumext.

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?.

12.40.1 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.

```
3640    \NewDocumentCommand \anspic { s o +m }
3641    {
```

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

```
3642      \bool_if:NF \l__enumext_store_active_bool
3643        {
3644          \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
3645        }
3646      \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3647        {
3648          \msg_error:nn { enumext } { keyanspic-wrong-level }
3649        }
3650      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
3651        {
3652          \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
3653        }
```

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.

```
3654      \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
3655        {
3656          \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
3657        }
3658    }
```

(*End of definition for* \anspic. *This function is documented on page* 16.)

__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.

```
3659    \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
3660    {
3661      \stepcounter { enumXvi }
3662      #3 \\
3663      \bool_if:nT { #1 }
3664        {
3665          \__enumext_keyans_addto_prop:n { #2 }
3666          \__enumext_keyans_store_ref:
3667          \__enumext_keyans_addto_seq:n { #2 }
3668          \int_gincr:N \g__enumext_check_starred_cmd_int
3669          \bool_lazy_or:nnT
3670            { \bool_if_p:N \l__enumext_show_answer_bool }
3671            { \bool_if_p:N \l__enumext_show_position_bool }
3672            {
3673              \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
3674              \__enumext_keyans_show_left:n { #2 }
3675              \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
3676            }
3677        }
3678      \tl_use:N \l__enumext_label_font_style_v_tl
3679      \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
3680    }
```

(*End of definition for* __enumext_keyans_anspic_code:nnn.)

12.40.2 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.

```
3681    \NewDocumentEnvironment{keyanspic}{ o }
3682    {
3683      \__enumext_keyans_pic_safe_exec:
3684      \__enumext_start_list:nn
3685        { }
3686        {
```

```
3687            \__enumext_keyans_pic_arg_two:
3688          }
```

We apply the "adjusted" vertical spacing above the environment

```
3689        \vspace { \l__enumext_keyans_pic_above_skip }
3690      }
```

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.

```
3691      {
3692        \tl_if_novalue:nTF { #1 }
3693          {
3694            \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
3695          }
3696          { \__enumext_keyans_pic_do:n { #1 } }
3697      \__enumext_stop_list:
3698      \__enumext_check_starred_cmd:n { anspic }
3699      \setcounter { enumXvi } { 0 }
3700      \vspace { \l__enumext_topsep_v_skip }
3701      %\bool_set_false:N \l__enumext_store_active_bool
3702      }
```

(*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.

```
3703  \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
3704    {
3705      \int_incr:N \l__enumext_keyans_pic_level_int
3706      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
3707        {
3708          \msg_error:nn { enumext } { keyanspic-nested }
3709        }
3710      \__enumext_keyans_name_and_start:
3711    }
```

(*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.

```
3712  \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3713    {
3714      \dim_compare:nNnT { #1 } < { 0pt }
3715        { \skip_set:Nn #1 { -#1 } }
3716    }
```

(*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.

```
3717  \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
3718    {
```

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.

```
3719      \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3720      \__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.

```
3721      \skip_add:Nn \parsep { \itemsep }
3722      \dim_add:Nn  \leftmargin { -\labelwidth - \labelsep }
3723      \dim_zero:N  \labelwidth
3724      \dim_zero:N  \listparindent
3725      \dim_zero:N  \labelsep
3726      \skip_zero:N \partopsep
3727      \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.

```
3728      \__enumext_keyans_pic_skip_abs:N \parsep
3729      \skip_set:Nn \l__enumext_keyans_pic_above_skip
3730        {
3731          \box_dp:N \strutbox
3732          + \l__enumext_topsep_v_skip
3733          - \parsep
3734        }
3735      \__enumext_item_std:w \scan_stop:
3736      % paranoia
3737      \RenewDocumentCommand \item {}
3738        {
3739          \msg_error:nn { enumext } { keyanspic-item-cmd }
3740        }
3741    }
```

(*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.

```
3742 \cs_new_protected:Nn \__enumext_keyans_pic_do:n
3743    {
3744      \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
3745    }
3746 \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.

```
3747 \cs_new_protected:Nn \__enumext_keyans_pic_row:n
3748    {
3749      \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
3750      \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
3751      \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
3752      \int_step_inline:nnn
3753        { \l__enumext_keyans_pic_above_int + 1 }
3754        { \l__enumext_keyans_pic_below_int }
3755        {
3756          \__enumext_minipage:w [ b ]{ \l__enumext_keyans_pic_width_dim }
3757            \centering
3758            \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
3759          \__enumext_endminipage:
3760        }
3761      \par
3762    }
```

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

12.41   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 (at least I have not achieved it) and as we will make it behave differently, we have no other option than to define a cascade of functions.

12.42   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.

```
3763 \cs_new_protected:Nn \__enumext_footnotetext:nn
3764    {
3765      \footnotetext[#1]{#2}
3766    }
3767 \cs_new_protected:Nn \__enumext_renew_footnote:
3768    {
```

```
3769    \seq_gclear:N \g__enumext_footnote_arg_seq
3770    \seq_gclear:N \g__enumext_footnote_int_seq
3771    \RenewDocumentCommand \footnote { o +m }
3772      {
3773        \tl_if_novalue:nTF {##1}
3774          {
3775            \stepcounter{footnote}
3776            \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
3777          }
3778          {
3779            \int_gset:Nn \g__enumext_footnote_int { ##1 }
3780          }
3781        \footnotemark [ \g__enumext_footnote_int ]
3782        \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
3783        \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
3784      }
3785    }
3786  \cs_new_protected:Nn \__enumext_print_footnote:
3787    {
3788      \seq_if_empty:NF \g__enumext_footnote_int_seq
3789        {
3790          \seq_map_pairwise_function:NNN
3791            \g__enumext_footnote_int_seq
3792            \g__enumext_footnote_arg_seq
3793            \__enumext_footnotetext:nn
3794        }
3795    }
```

(*End of definition for* `\__enumext_footnotetext:nn`, `\__enumext_renew_footnote:`, *and* `\__enumext_print_footnote:`.)

12.42.1 Functions for item box width

To achieve the horizontal list environment we will capture the `\item` command and the content of this in an plain `lrbox` 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.

```
3796  \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
3797    {
3798      \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3799        {
3800          \dim_set:Nn \l__enumext_columns_sep_vii_dim
3801            {
3802              ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
3803              / \l__enumext_columns_vii_int
3804            }
3805        }
3806      \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - 1 }
3807      \dim_set:Nn \l__enumext_item_width_vii_dim
3808        {
3809          ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
3810          / \l__enumext_columns_vii_int
3811          - \l__enumext_labelwidth_vii_dim
3812          - \l__enumext_labelsep_vii_dim
3813        }
```

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

```
3814        \dim_compare:nNnT { \l__enumext_rightmargin_vii_dim } > { \c_zero_dim }
3815          {
3816            \dim_sub:Nn \l__enumext_item_width_vii_dim
3817              {
3818                ( \l__enumext_rightmargin_vii_dim * \l__enumext_tmpa_vii_int )
3819                / \l__enumext_columns_vii_int
3820              }
3821            \dim_add:Nn \l__enumext_columns_sep_vii_dim
3822              {
3823                \l__enumext_rightmargin_vii_dim
3824              }
3825          }
3826    }
```

Same implementation for the `keyans*` environment.

```
3827 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3828   {
3829     \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3830       {
3831         \dim_set:Nn \l__enumext_columns_sep_viii_dim
3832           {
3833             ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
3834             / \l__enumext_columns_viii_int
3835           }
3836       }
3837     \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - 1 }
3838     \dim_set:Nn \l__enumext_item_width_viii_dim
3839       {
3840         ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
3841         / \l__enumext_columns_viii_int
3842         - \l__enumext_labelwidth_viii_dim
3843         - \l__enumext_labelsep_viii_dim
3844       }
3845     \dim_compare:nNnT { \l__enumext_rightmargin_viii_dim } > { \c_zero_dim }
3846       {
3847         \dim_sub:Nn \l__enumext_item_width_viii_dim
3848           {
3849             ( \l__enumext_rightmargin_viii_dim * \l__enumext_tmpa_vii_int )
3850             / \l__enumext_columns_viii_int
3851           }
3852         \dim_add:Nn \l__enumext_columns_sep_viii_dim
3853           {
3854             \l__enumext_rightmargin_viii_dim
3855           }
3856       }
3857   }
```

(*End of definition for* `\__enumext_starred_columns_set_vii:` *and* `\__enumext_starred_columns_set_viii:`.)

12.42.2 Functions for join item columns

`\__enumext_starred_joined_item_vii:n`
`\__enumext_starred_joined_item_viii:n`

The functions `\__enumext_starred_joined_item_vii:n` and `\__enumext_starred_joined_item_-viii:n` will set the *width* of the box in which the content passed to `\item`(⟨*columns*⟩) will be stored together with the value of `\itemwidth` for the `enumext*` environment.

```
3858 \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
3859   {
3860     \int_set:Nn \l__enumext_joined_item_vii_int {#1}
3861     \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
3862       {
3863         \msg_warning:nnee { enumext } { item-joined }
3864           { \int_use:N \l__enumext_joined_item_vii_int }
3865           { \int_use:N \l__enumext_columns_vii_int }
3866         \int_set:Nn \l__enumext_joined_item_vii_int
3867           {
3868             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
3869           }
3870       }
3871     \int_compare:nNnT
3872       { \l__enumext_joined_item_vii_int }
3873       >
3874       { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
3875       {
3876         \msg_warning:nnee { enumext } { item-joined-columns }
3877           { \int_use:N \l__enumext_joined_item_vii_int }
3878           {
3879             \int_eval:n
3880               { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
3881           }
3882         \int_set:Nn \l__enumext_joined_item_vii_int
3883           {
3884             \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
3885           }
3886       }
3887     \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { 1 }
3888       {
3889         \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
```

```
3890        \int_decr:N \l__enumext_joined_item_aux_vii_int
3891        \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3892        \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3893        \dim_set:Nn \l__enumext_joined_width_vii_dim
3894          {
3895            \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
3896          + (   \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3897            + \l__enumext_columns_sep_vii_dim
3898          )*\l__enumext_joined_item_aux_vii_int
3899          }
3900        \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
3901      }
3902      {
3903        \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
3904        \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
3905      }
3906    }
```

Same implementation for the keyans* environment.

```
3907  \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
3908    {
3909      \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3910      \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3911        {
3912          \msg_warning:nnee { enumext } { item-joined }
3913            { \int_use:N \l__enumext_joined_item_viii_int }
3914            { \int_use:N \l__enumext_columns_viii_int }
3915          \int_set:Nn \l__enumext_joined_item_viii_int
3916            {
3917              \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
3918            }
3919        }
3920      \int_compare:nNnT
3921        { \l__enumext_joined_item_viii_int }
3922        >
3923        { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
3924        {
3925          \msg_warning:nnee { enumext } { item-joined-columns }
3926            { \int_use:N \l__enumext_joined_item_viii_int }
3927            {
3928              \int_eval:n
3929                { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
3930            }
3931          \int_set:Nn \l__enumext_joined_item_viii_int
3932            {
3933              \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
3934            }
3935        }
3936      \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { 1 }
3937        {
3938          \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3939          \int_decr:N \l__enumext_joined_item_aux_viii_int
3940          \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
3941          \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
3942          \dim_set:Nn \l__enumext_joined_width_viii_dim
3943            {
3944              \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3945            + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3946              + \l__enumext_columns_sep_viii_dim
3947            )*\l__enumext_joined_item_aux_viii_int
3948            }
3949          \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3950        }
3951        {
3952          \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3953          \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3954        }
3955    }
```

(*End of definition for* __enumext_starred_joined_item_vii:n *and* __enumext_starred_joined_item_viii:n.)

12.42.3 Functions for `mini-env`, `mini-right` and `mini-right*` keys

`\__enumext_start_mini_vii:`
`\__enumext_stop_mini_vii:`

The implementation of the `mini-env` key support is almost identical to the one used in the `enumext` and `keyans` environments, the difference is that the `__enumext_mini_env*` 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`.

```
3956 \cs_new_protected:Nn \__enumext_start_mini_vii:
3957   {
3958     \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
3959       {
3960         \dim_set:Nn \l__enumext_minipage_left_vii_dim
3961           {
3962             \linewidth
3963             - \l__enumext_minipage_right_vii_dim
3964             - \l__enumext_minipage_hsep_vii_dim
3965           }
3966         \bool_set_true:N \l__enumext_minipage_active_vii_bool
3967         \dim_gset_eq:NN
3968           \g__enumext_minipage_right_vii_dim
3969           \l__enumext_minipage_right_vii_dim
3970         \__enumext_mini_addvspace_vii:
3971         \nointerlineskip\noindent
3972         \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
3973       }
3974   }
```

The function `\__enumext_stop_mini_vii:` closes the `__enumext_mini_env*` 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_-env*` on the *"right side"*.

```
3975 \cs_new_protected:Nn \__enumext_stop_mini_vii:
3976   {
3977     \bool_if:NT \l__enumext_minipage_active_vii_bool
3978       {
3979         \end{__enumext_mini_env*}
3980         \hfill
3981         \bool_gset_true:N \g__enumext_minipage_active_vii_bool
3982       }
3983   }
```

Finally we execute the ${\langle code \rangle}$ passed to the `mini-right` or `mini-right*` keys stored in the variable `\g__enumext_miniright_code_vii_tl` in the `__enumext_mini_env*` environment on the *"right side"*. For compatibility with the `caption` package and possibly other ${\langle code \rangle}$ passed to this key, we will pass it to a box and then print it.

```
3984 \__enumext_after_env:nn {enumext*}
3985   {
3986     \bool_if:NT \g__enumext_minipage_active_vii_bool
3987       {
3988         \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
3989           \par\addvspace { \g__enumext_minipage_right_skip }
3990           \bool_if:NF \g__enumext_minipage_center_vii_bool
3991             {
3992               \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
3993                 {
3994                   \centering
3995                 }
3996             }
3997           \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
3998             {
3999               \tl_use:N \g__enumext_miniright_code_vii_tl
4000             }
4001           \box_use_drop:N \l__enumext_miniright_code_vii_box
4002         \end{__enumext_mini_env*}
4003         \par\addvspace{ \g__enumext_minipage_after_skip }
4004       }
4005     \bool_gset_false:N \g__enumext_minipage_active_vii_bool
4006     \bool_gset_true:N \g__enumext_minipage_center_vii_bool
4007     \tl_gclear:N \g__enumext_miniright_code_vii_tl
4008     \dim_gzero:N \g__enumext_minipage_right_vii_dim
4009     \bool_gset_false:N \g__enumext_starred_bool
4010   }
```

(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.

```
4011  \cs_new_protected:Nn \__enumext_start_mini_viii:
4012    {
4013      \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
4014        {
4015          \dim_set:Nn \l__enumext_minipage_left_viii_dim
4016            {
4017              \linewidth
4018              - \l__enumext_minipage_right_viii_dim
4019              - \l__enumext_minipage_hsep_viii_dim
4020            }
4021          \bool_set_true:N \l__enumext_minipage_active_viii_bool
4022          \dim_gset_eq:NN
4023            \g__enumext_minipage_right_viii_dim
4024            \l__enumext_minipage_right_viii_dim
4025          \__enumext_mini_addvspace_viii:
4026          \nointerlineskip\noindent
4027          \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
4028        }
4029    }
4030  \cs_new_protected:Nn \__enumext_stop_mini_viii:
4031    {
4032      \bool_if:NT \l__enumext_minipage_active_viii_bool
4033        {
4034          \end{__enumext_mini_env*}
4035          \hfill
4036          \bool_gset_true:N \g__enumext_minipage_active_viii_bool
4037        }
4038    }
4039  \__enumext_after_env:nn {keyans*}
4040    {
4041      \bool_if:NT \g__enumext_minipage_active_viii_bool
4042        {
4043          \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
4044            \par\addvspace { \g__enumext_minipage_right_skip }
4045            \bool_if:NF \g__enumext_minipage_center_viii_bool
4046              {
4047                \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
4048                  {
4049                    \centering
4050                  }
4051              }
4052            \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
4053              {
4054                \tl_use:N \g__enumext_miniright_code_viii_tl
4055              }
4056            \box_use_drop:N \l__enumext_miniright_code_viii_box
4057          \end{__enumext_mini_env*}
4058          \par\addvspace{ \g__enumext_minipage_after_skip }
4059        }
4060      \bool_gset_false:N \g__enumext_minipage_active_viii_bool
4061      \bool_gset_true:N \g__enumext_minipage_center_viii_bool
4062      \tl_gclear:N \g__enumext_miniright_code_viii_tl
4063      \dim_gzero:N \g__enumext_minipage_right_viii_dim
4064    }
```

(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 `\noindent` and next to `\item` equal to `\__enumext_start_item_tmp_vii:` which we will redefine later.

```
4065  \NewDocumentEnvironment{enumext*}{ o }
4066    {
4067      \__enumext_safe_exec_vii:
4068      \__enumext_parse_keys_vii:n {#1}
4069      \__enumext_before_list_vii:
```

```
4070    \__enumext_start_store_level_vii:
4071    \__enumext_start_list:nn { }
4072      {
4073        \__enumext_list_arg_two_vii:
4074        \__enumext_before_keys_exec_vii:
4075      }
4076    \__enumext_starred_columns_set_vii:
4077    \item[] \scan_stop:
4078    \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
4079    \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
4080  }
4081  {
4082    \__enumext_stop_item_tmp_vii:
4083    \__enumext_remove_extra_parsep_vii:
4084    \__enumext_stop_list:
4085    \__enumext_stop_store_level_vii:
4086    \__enumext_after_list_vii:
4087  }
```

(*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_env*, 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.

```
4088  \cs_new_protected:Nn \__enumext_safe_exec_vii:
4089    {
4090      \__enumext_internal_mini_page:
4091      \__enumext_is_not_nested:
4092      \int_incr:N \l__enumext_level_h_int
4093      \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
4094        {
4095          \msg_error:nn { enumext } { nested }
4096        }
4097      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
4098        {
4099          \msg_error:nnn { enumext } { nested-horizontal } { keyans*}
4100        }
4101      \bool_set_true:N \l__enumext_starred_bool
4102      \bool_set_false:N \l__enumext_standar_bool
4103      \__enumext_is_on_first_level:
4104    }
```

(*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 envi-
ronment [⟨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.

```
4105  \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
4106    {
4107      \tl_if_novalue:nF {#1}
4108        {
4109          \str_clear:N \l__enumext_series_str
4110          \keys_set:nn { enumext / enumext* } {#1}
4111          \__enumext_parse_series:n {#1}
4112          \__enumext_store_active_keys_vii:n {#1}
4113          \__enumext_nested_base_line_fix:
4114        }
4115    }
```

(*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*.

```
4116  \cs_new_protected:Nn \__enumext_before_list_vii:
4117    {
4118      \__enumext_vspace_above_vii:
4119      \__enumext_check_ans_active:
4120      \__enumext_before_args_exec_vii:
4121      \__enumext_start_mini_vii:
4122    }
```

(*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.

```
4123  \cs_new_protected:Nn \__enumext_after_list_vii:
4124    {
4125      \__enumext_stop_mini_vii:
4126      \__enumext_after_stop_list_vii:
4127      \__enumext_check_ans_key_hook:
4128      \__enumext_vspace_below_vii:
4129      \bool_set_false:N \l__enumext_starred_bool
4130      \__enumext_resume_save_counter:
4131    }
```

(*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
__enumext_stop_store_level_vii: activate the level saving mechanism for storage in ⟨*sequence*⟩ of the \anskey command and anskey* environment if enumext* are nested in enumext.

```
4132  \cs_new_protected:Nn \__enumext_start_store_level_vii:
4133    {
4134      \bool_if:NT \l__enumext_store_active_bool
4135        {
4136          \int_compare:nNnT { \l__enumext_level_int } > { 0 }
4137            {
4138              \__enumext_store_level_open_vii:
4139            }
4140        }
4141    }
4142  \cs_new_protected:Nn \__enumext_stop_store_level_vii:
4143    {
4144      \bool_if:NT \l__enumext_store_active_bool
4145        {
4146          \int_compare:nNnT { \l__enumext_level_int } > { 0 }
4147            {
4148              \__enumext_store_level_close_vii:
4149            }
4150        }
4151    }
```

(*End of definition for* __enumext_start_store_level_vii: *and* __enumext_stop_store_level_vii:.)

12.43.1 The command \item in enumext*

__enumext_start_item_tmp_vii: First we will call the function __enumext_stop_item_tmp_vii: that we will redefine later, we will increment the value of \l__enumext_item_column_pos_vii_int that will count the item's by rows and the value of \g__enumext_item_count_all_vii_int that will count the total of item's in the environment. After that we will call the function __enumext_item_peek_args_vii: that will handle the arguments passed to \item.

```
4152  \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
4153    {
4154      \__enumext_stop_item_tmp_vii:
4155      \int_incr:N \l__enumext_item_column_pos_vii_int
4156      \int_gincr:N \g__enumext_item_count_all_vii_int
4157      \__enumext_item_peek_args_vii:
4158    }
```

(*End of definition for* __enumext_start_item_tmp_vii:.)

__enumext_item_peek_args_vii:

The function __enumext_item_peek_args_vii: will handle the \item(⟨*number*⟩). Look for the argument "(", if it is present we will call the function __enumext_joined_item_vii:w (⟨*number*⟩), which is in charge of joining the item's in the same row, in case they are not present we will set the default value (1).

```
4159 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
4160    {
4161       \peek_meaning:NTF (
4162          { \__enumext_joined_item_vii:w }
4163          { \__enumext_joined_item_vii:w (1) }
4164    }
```

(*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.

```
4165 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
4166    {
4167       \__enumext_starred_joined_item_vii:n {#1}
4168       \peek_meaning_remove:NTF *
4169          { \__enumext_starred_item_vii:w  }
4170          { \__enumext_standar_item_vii:w }
4171    }
```

(*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].

```
4172 \cs_new_protected:Npn \__enumext_standar_item_vii:w
4173    {
4174       \bool_set_false:N \l__enumext_item_starred_vii_bool
4175          \peek_meaning:NTF [
4176             {
4177                \bool_set_eq:NN
4178                   \l__enumext_wrap_label_vii_bool
4179                   \l__enumext_wrap_label_opt_vii_bool
4180                \__enumext_start_item_vii:w
4181             }
4182             {
4183                \bool_set_true:N \l__enumext_wrap_label_vii_bool
4184                \legacy_if_set_true:n { @noitemarg }
4185                \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4186             }
4187    }
```

(*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*⟩].

```
4188 \cs_new_protected:Npn \__enumext_starred_item_vii:w
4189    {
4190       \bool_set_true:N \l__enumext_item_starred_vii_bool
4191       \bool_set_true:N \l__enumext_wrap_label_vii_bool
4192       \peek_meaning:NTF [
4193          { \__enumext_starred_item_vii_aux_i:w }
4194          { \__enumext_starred_item_vii_aux_ii:w }
4195    }
4196 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
4197    {
4198       \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
4199       \__enumext_starred_item_vii_aux_ii:w
4200    }
4201 \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
```

```
4202  {
4203    \peek_meaning:NTF [
4204      { \__enumext_starred_item_vii_aux_iii:w }
4205      {
4206        \dim_set_eq:NN
4207          \l__enumext_item_symbol_sep_vii_dim
4208          \l__enumext_labelsep_vii_dim
4209        \legacy_if_set_true:n { @noitemarg }
4210        \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4211      }
4212  }
4213  \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
4214    {
4215      \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
4216      \legacy_if_set_true:n { @noitemarg }
4217      \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4218    }
```

(*End of definition for __enumext_starred_item_vii:w and others.*)

12.43.2   Real definition of \item **in enumext***

__enumext_start_item_vii:w   The functions __enumext_start_item_vii:w and __enumext_stop_item_vii: executing the true
definition of \item inside the enumext* environment. 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
and add the hyperref compatible enumXvii counter, after that we will start capturing the item content in
a box. Here need setting the \if@hyper@item switch to *"true"* for hyperref compatible. The explanation
for this is given by the master Heiko Oberdiek on \refstepcounter{enumi} twice (or more) creates destination
with the same identifier.

```
4219  \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
4220    {
4221      \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
4222      \legacy_if:nT { @noitemarg }
4223      {
4224        \legacy_if_set_false:n { @noitemarg }
4225        \legacy_if:nT { @nmbrlist }
4226          {
4227            \bool_if:NT \l__enumext_hyperref_bool
4228              {
4229                \legacy_if_set_true:n { @hyper@item }
4230              }
4231            \refstepcounter{enumXvii}
4232            \bool_if:NT \l__enumext_check_answers_bool
4233              {
4234                \int_gincr:N \g__enumext_item_number_int
4235                \bool_set_true:N \l__enumext_item_number_bool
4236              }
4237          }
4238      }
```

Here we start capturing \item and its contents into a group using the plain form of the lrbox environment.
If the state of the variable \l__enumext_footnotes_key_bool is false, we will redefine the command
\footnote, followed by printing the *⟨symbol⟩* defined for \item* if it is present and open a new group
inside which we execute font key next to \item and the keys wrap-label, wrap-label*, align,
close the group and execute the key labelsep and then the key first. Finally we open the minipage
environment and execute the listparindent key which will be equal to \parindent, the parsep key
which will be equal to \parskip and the itemindent key.

```
4239        \group_begin:
4240          \lrbox{ \l__enumext_item_text_vii_box }
4241          \bool_if:NF \l__enumext_footnotes_key_bool
4242            {
4243              \__enumext_renew_footnote:
4244            }
4245          \bool_if:NT \l__enumext_item_starred_vii_bool
4246            {
4247              \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
4248                {
4249                  \tl_gset_eq:NN
4250                    \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
4251                }
4252              \mode_leave_vertical:
```

```
4253          \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
4254          \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
4255          \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
4256          \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
4257        }
4258      \group_begin:
4259        \tl_use:N \l__enumext_label_font_style_vii_tl
4260        \bool_if:NTF \l__enumext_wrap_label_vii_bool
4261          {
4262            \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
4263              { \__enumext_wrapper_label_vii:n {#1} }
4264          }
4265          {
4266            \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
4267          }
4268      \group_end:
4269      \skip_horizontal:N \l__enumext_labelsep_vii_dim
4270      \tl_use:N \l__enumext_after_list_args_vii_tl
4271      \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim  }
4272        \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
4273        \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
4274        \tl_use:N \l__enumext_fake_item_indent_vii_tl
4275    }
```

(*End of definition for* __enumext_start_item_vii:w.)

__enumext_stop_item_vii: The function __enumext_stop_item_vii: shall terminate with the capture of \item and its ⟨contents⟩. Close the environments minipage, lrbox and the group. Then we only have to set the width of the box and print it next to \footnote, and add the horizontal and vertical separation between the boxes.

```
4276 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
4277   {
4278        \__enumext_endminipage:
4279      \endlrbox
4280    \group_end:
4281    \box_set_wd:Nn \l__enumext_item_text_vii_box
4282      {
4283        \l__enumext_joined_width_vii_dim
4284        + \l__enumext_labelwidth_vii_dim
4285        + \l__enumext_labelsep_vii_dim
4286      }
4287    \int_set:Nn \hbadness { 10000 }
4288    \box_use_drop:N \l__enumext_item_text_vii_box
4289    \bool_if:NF \l__enumext_footnotes_key_bool
4290      {
4291        \__enumext_print_footnote:
4292      }
4293    \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
4294      {
4295        \par\noindent
4296        \int_zero:N \l__enumext_item_column_pos_vii_int
4297      }
4298      { \hspace{ \l__enumext_columns_sep_vii_dim } }
4299   }
```

(*End of definition for* __enumext_stop_item_vii:.)

__enumext_remove_extra_parsep_vii: 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.

```
4300 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
4301   {
4302    \int_compare:nNnT
4303      {
4304        \int_mod:nn {  \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
4305      }
4306      =
4307      { 0 }
4308      {
4309        \par
4310        \vspace{ -\l__enumext_itemsep_vii_skip }
4311        \int_gzero:N \g__enumext_item_count_all_vii_int
4312      }
4313   }
```

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.

```
4314 \__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 \noindent and next to \item equal to __enumext_start_item_tmp_-viii: which we will redefine later.

```
4315 \NewDocumentEnvironment{keyans*}{ o }
4316   {
4317     \__enumext_safe_exec_viii:
4318     \__enumext_parse_keys_viii:n {#1}
4319     \__enumext_before_list_viii:
4320     \__enumext_start_list:nn { }
4321       {
4322         \__enumext_list_arg_two_viii:
4323         \__enumext_before_keys_exec_viii:
4324       }
4325     \__enumext_starred_columns_set_viii:
4326     \item[] \scan_stop:
4327     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
4328     \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
4329   }
4330   {
4331     \__enumext_stop_item_tmp_viii:
4332     \__enumext_remove_extra_parsep_viii:
4333     \__enumext_check_starred_cmd:n { item }
4334     \__enumext_stop_list:
4335     \__enumext_after_list_viii:
4336   }
```

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

__enumext_safe_exec_viii: First check the maximum nesting level for the keyans* environment.

```
4337 \cs_new_protected:Nn \__enumext_safe_exec_viii:
4338   {
4339     \int_incr:N \l__enumext_keyans_level_h_int
4340     \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
4341       {
4342         \msg_error:nn { enumext } { nested }
4343       }
4344     \__enumext_keyans_name_and_start:
4345     \bool_if:NT \l__enumext_starred_bool
4346       {
4347         \msg_error:nnn { enumext } { nested-horizontal } { enumext* }
4348       }
4349     \bool_set_true:N \l__enumext_starred_bool
4350     % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
4351     \bool_set_false:N \l__enumext_store_active_bool
4352     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
4353       {
4354         \msg_error:nn { enumext } { keyans-wrong-level }
4355       }
4356   }
```

(*End of definition for* __enumext_safe_exec_viii:.)

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

```
4357 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
4358   {
4359     \tl_if_novalue:nF {#1}
4360       {
4361         \keys_set:nn { enumext / keyans* } {#1}
4362       }
4363   }
```

(*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.

```
4364  \cs_new_protected:Nn \__enumext_before_list_viii:
4365    {
4366      \__enumext_vspace_above_viii:
4367      \__enumext_before_args_exec_viii:
4368      \__enumext_start_mini_viii:
4369    }
```

(*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.

```
4370  \cs_new_protected:Nn \__enumext_after_list_viii:
4371    {
4372      \__enumext_stop_mini_viii:
4373      \__enumext_after_stop_list_viii:
4374      \__enumext_vspace_below_viii:
4375    }
```

(*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_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.

```
4376  \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
4377    {
4378      \__enumext_stop_item_tmp_viii:
4379      \int_incr:N \l__enumext_item_column_pos_viii_int
4380      \int_gincr:N \g__enumext_item_count_all_viii_int
4381      \__enumext_item_peek_args_viii:
4382    }
```

(*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).

```
4383  \cs_new_protected:Nn \__enumext_item_peek_args_viii:
4384    {
4385      \peek_meaning:NTF (
4386        { \__enumext_joined_item_viii:w }
4387        { \__enumext_joined_item_viii:w (1) }
4388    }
```

(*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.

```
4389  \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
4390    {
4391      \__enumext_starred_joined_item_viii:n {#1}
4392      \peek_meaning_remove:NTF *
4393        { \__enumext_starred_item_viii:w  }
4394        { \__enumext_standar_item_viii:w }
4395    }
```

(*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].

```
4396  \cs_new_protected:Npn \__enumext_standar_item_viii:w
4397    {
4398      \bool_set_false:N \l__enumext_item_starred_viii_bool
4399        \peek_meaning:NTF [
4400          {
4401            \bool_set_eq:NN
4402              \l__enumext_wrap_label_viii_bool
4403              \l__enumext_wrap_label_opt_viii_bool
4404            \__enumext_start_item_viii:w
4405          }
4406          {
4407            \bool_set_true:N \l__enumext_wrap_label_viii_bool
4408            \legacy_if_set_true:n { @noitemarg }
4409            \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4410          }
4411    }
```

(*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*⟩].

```
4412  \cs_new_protected:Npn \__enumext_starred_item_viii:w
4413    {
4414      \bool_set_true:N \l__enumext_item_starred_viii_bool
4415      \bool_set_true:N \l__enumext_wrap_label_viii_bool
4416      \peek_meaning:NTF [
4417        { \__enumext_starred_item_viii_aux_i:w }
4418        { \__enumext_starred_item_viii_aux_ii:w }
4419    }
```

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.

```
4420  \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
4421    {
4422      \tl_clear:N \l__enumext_store_current_label_tl
4423      \tl_if_novalue:nF { #1 }
4424        {
4425          \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
4426            {
4427              \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_op
4428              \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
4429            }
4430          \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
4431        }
4432      \__enumext_starred_item_viii_aux_ii:w
4433    }
4434  \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
4435    {
4436      \legacy_if_set_true:n { @noitemarg }
4437      \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4438    }
```

(*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.

```
4439  \cs_new_protected:Nn \__enumext_starred_item_exec:
4440    {
4441      \tl_put_left:Ne \l__enumext_store_current_label_tl { \l__enumext_label_viii_tl }
4442      \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
4443      \__enumext_keyans_store_ref:
4444      \tl_put_left:Ne \l__enumext_store_current_label_tl { \item }
4445      \__enumext_keyans_addto_seq_link:
4446      \int_gincr:N \g__enumext_check_starred_cmd_int
4447      \bool_if:NT \l__enumext_show_answer_bool
4448        {
4449          \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4450        }
4451      \bool_if:NT \l__enumext_show_position_bool
4452        {
4453          \tl_set:Ne \l__enumext_mark_answer_sym_tl
4454            {
4455              \group_begin:
4456                \exp_not:N \normalfont
4457                \exp_not:N \footnotesize [ \int_eval:n
4458                  {
4459                    \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
4460                  }
4461                ]
4462              \group_end:
4463            }
4464          \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4465        }
4466    }
```

(*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.

```
4467  \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
4468    {
4469      \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
4470      \legacy_if:nT { @noitemarg }
4471        {
4472          \legacy_if_set_false:n { @noitemarg }
4473          \legacy_if:nT { @nmbrlist }
4474            {
4475              \bool_if:NT \l__enumext_hyperref_bool
4476                {
4477                  \legacy_if_set_true:n { @hyper@item }
4478                }
4479              \refstepcounter{enumXviii}
4480            }
4481        }
```

Here we start capturing `\item` and its contents into a group using the plain form of the `lrbox` environment.

```
4482      \group_begin:
4483        \lrbox{ \l__enumext_item_text_viii_box }
4484          \bool_if:NF \l__enumext_footnotes_key_bool
4485            {
4486              \__enumext_renew_footnote:
4487            }
4488          \bool_if:NT \l__enumext_item_starred_viii_bool
4489            {
4490              \__enumext_starred_item_exec:
4491            }
4492          \group_begin:
4493            \tl_use:N \l__enumext_label_font_style_viii_tl
4494            \bool_if:NTF \l__enumext_wrap_label_viii_bool
4495              {
4496                \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
4497                  { \__enumext_wrapper_label_viii:n {#1} }
4498              }
4499              {
4500                \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1
```

```
4501                }
4502            \group_end:
4503            \skip_horizontal:N \l__enumext_labelsep_viii_dim
4504            \tl_use:N \l__enumext_after_list_args_viii_tl
4505            \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim  }
4506              \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
4507              \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
4508              \bool_if:NT \l__enumext_item_starred_viii_bool
4509                {
4510                   \tl_use:N \l__enumext_fake_item_indent_viii_tl
4511                   \__enumext_keyans_show_item_opt:
4512                   \skip_horizontal:n { -\l__enumext_fake_item_indent_viii_dim - \l__enumext_labelsep_ᵥ
4513                }
4514                {
4515                   \tl_use:N \l__enumext_fake_item_indent_viii_tl
4516                }
4517        }
```

(*End of definition for* `\__enumext_start_item_viii:w`.)

`\__enumext_stop_item_viii:` The function `\__enumext_stop_item_viii:` shall terminate with the capture of `\item` and its ⟨*contents*⟩. Close the environments `minipage`, `lrbox` and the group. Then we only have to set the width of the box and print it next to `\footnote`, and add the horizontal and vertical separation between the boxes.

```
4518  \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
4519    {
4520            \__enumext_endminipage:
4521          \endlrbox
4522        \group_end:
4523        \box_set_wd:Nn \l__enumext_item_text_viii_box
4524          {
4525             \l__enumext_joined_width_viii_dim
4526             + \l__enumext_labelwidth_viii_dim
4527             + \l__enumext_labelsep_viii_dim
4528          }
4529        \int_set:Nn \hbadness { 10000 }
4530        \box_use_drop:N \l__enumext_item_text_viii_box
4531        \bool_if:NF \l__enumext_footnotes_key_bool
4532          {
4533             \__enumext_print_footnote:
4534          }
4535        \int_compare:nNnTF
4536          { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
4537          {
4538             \par\noindent
4539             \int_zero:N \l__enumext_item_column_pos_viii_int
4540          }
4541          { \hspace{ \l__enumext_columns_sep_viii_dim } }
4542    }
```

(*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.

```
4543  \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
4544    {
4545      \int_compare:nNnT
4546        {
4547           \int_mod:nn
4548             { \g__enumext_item_count_all_viii_int }
4549             { \l__enumext_columns_viii_int }
4550        }
4551        =
4552        { 0 }
4553        {
4554           \par
4555           \vspace{ -\l__enumext_itemsep_viii_skip }
4556           \int_gzero:N \g__enumext_item_count_all_viii_int
4557        }
4558    }
```

(*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.

```
4559  \NewDocumentCommand \getkeyans { m }
4560    {
4561      \exp_args:Ne \__enumext_getkeyans_aux:n
4562        { \tl_to_str:e { \text_expand:n {#1} } }
4563    }
```

(*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.

```
4564  \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
4565    {
4566      \str_if_in:nnTF {#1} { : }
4567        {
4568          \use:e
4569            {
4570              \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
4571                { {##1} {##2} }
4572            }
4573          \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
4574        }
4575        { \msg_error:nnn { enumext } { missing-colon } {#1} }
4576    }
```

(*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*⟩.

```
4577  \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
4578    {
4579      \prop_if_exist:cF { g__enumext_#1_prop }
4580        { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
4581      \group_begin:
4582        \prop_item:cn { g__enumext_#1_prop }{#2}
4583      \group_end:
4584    }
```

(*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*⟩].

```
4585  \keys_define:nn  { enumext / print }
4586    {
4587      print*  .code:n     = \keys_precompile:neN { enumext / enumext* }
4588                             { \__enumext_filter_save_key:n {#1} }
4589                             \l__enumext_print_keyans_starred_tl, % starred cmd
4590      print*  .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4591      print-1 .code:n     = \keys_precompile:neN { enumext / level-1 }
4592                             { \__enumext_filter_save_key:n {#1} }
4593                             \l__enumext_print_keyans_i_tl,
4594      print-1 .initial:n = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4595      print-2 .code:n     = \keys_precompile:neN { enumext / level-2 }
4596                             { \__enumext_filter_save_key:n {#1} }
4597                             \l__enumext_print_keyans_ii_tl,
4598      print-2 .initial:n = { nosep, label=(\alph*), first=\small, font=\small },
4599      print-3 .code:n     = \keys_precompile:neN { enumext / level-3 }
4600                             { \__enumext_filter_save_key:n {#1} }
```

```
4601                                          \l__enumext_print_keyans_iii_tl,
4602        print-3 .initial:n  = { nosep, label=\roman*., first=\small, font=\small },
4603        print-4 .code:n     = \keys_precompile:neN { enumext / level-4 }
4604                                 { \__enumext_filter_save_key:n {#1} }
4605                              \l__enumext_print_keyans_iv_tl,
4606        print-4 .initial:n  = { nosep, label=\Alph*., first=\small, font=\small },
4607        print-* .code:n     = \keys_precompile:neN { enumext / enumext* }
4608                                 { \__enumext_filter_save_key:n {#1} }
4609                              \l__enumext_print_keyans_vii_tl, % starred nested
4610        print-* .initial:n  = { nosep, label=\arabic*., first=\small, font=\small },
4611      }
```

💣 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.

```
4612  \NewDocumentCommand \printkeyans { s O{} m }
4613    {
4614      \group_begin:
4615        \tl_use:N \l__enumext_print_keyans_i_tl
4616        \tl_use:N \l__enumext_print_keyans_ii_tl
4617        \tl_use:N \l__enumext_print_keyans_iii_tl
4618        \tl_use:N \l__enumext_print_keyans_iv_tl
4619        \tl_use:N \l__enumext_print_keyans_vii_tl
4620        \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
4621      \group_end:
4622    }
```

(*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.

```
4623  \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
4624    {
4625      \seq_if_exist:cTF { g__enumext_#3_seq }
4626        {
4627          \seq_if_empty:cF { g__enumext_#3_seq }
4628            {
4629              %%\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⟩.

```
4630              \bool_if:nTF {#1}
4631                {
4632                  \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
4633                    {
4634                      \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
4635                    }
4636                    {
4637                      \tl_use:N \l__enumext_print_keyans_starred_tl
4638                      \begin{enumext*}[#2]
4639                        \keys_set:nn { enumext / level-1 }{ base-fix }
4640                        \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4641                      \end{enumext*}
4642                    }
4643                }
```

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⟩.

```
4644                {
4645                  \begin{enumext}[#2]
4646                    \keys_set:nn { enumext / enumext* }{ base-fix }
4647                    \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4648                  \end{enumext}
4649                }
4650            }
```

```
4651              }
4652           {
4653              \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
4654           }
4655    }
```

(*End of definition for* __enumext_printkeyans:nnn.)

12.47   The command \setenumext

The command \setenumext will be in charge of managing the ⟨*keys*⟩ passed to all environments and to the \printkeyans command. We must take precautions with the enumext* environment and *"first level"* of the enumext environment so as not to capture ⟨*keys*⟩ that complicate us.

__enumext_filter_first_level:n
__enumext_filter_first_level_key:n
__enumext_filter_first_level_pair:nn

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

```
4656  \cs_new:Npn \__enumext_filter_first_level:n #1
4657    {
4658       \use:e
4659          {
4660             \keyval_parse:NNn
4661                \__enumext_filter_first_level_key:n
4662                \__enumext_filter_first_level_pair:nn {#1}
4663          }
4664    }
```

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*.

```
4665  \cs_new:Npn \__enumext_filter_first_level_key:n #1
4666    {
4667       \str_case:nnF {#1}
4668          {
4669             { resume   } {}
4670             { resume*  } {}
4671          }
4672          { , { \exp_not:n {#1} } }
4673    }
```

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.

```
4674  \cs_new:Npn \__enumext_filter_first_level_pair:nn #1#2
4675    {
4676       \str_case:nnF {#1}
4677          {
4678             { series } {}
4679             { resume } {}
4680             { save-ans } {}
4681          }
4682          { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
4683    }
```

(*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.

```
4684  \keys_define:nn { enumext / meta-families }
4685    {
4686       enumext-1 .code:n =
4687                  {
4688                     \keys_set:ne { enumext / level-1  }
4689                        {
4690                           \__enumext_filter_first_level:n {#1}
4691                        }
4692                  } ,
4693       enumext-2 .code:n = { \keys_set:nn { enumext / level-2  } {#1} } ,
4694       enumext-3 .code:n = { \keys_set:nn { enumext / level-3  } {#1} } ,
4695       enumext-4 .code:n = { \keys_set:nn { enumext / level-4  } {#1} } ,
4696       keyans    .code:n = { \keys_set:nn { enumext / keyans   } {#1} } ,
4697       enumext*  .code:n =
4698                  {
4699                     \keys_set:ne { enumext / enumext* }
4700                        {
4701                           \__enumext_filter_first_level:n {#1}
```

```
4702                         }
4703                   } ,
4704      keyans*    .code:n = { \keys_set:nn { enumext / keyans*  } {#1} } ,
4705      print*     .code:n = { \keys_set:nn { enumext / print     } { print*  = {#1} } } ,
4706      print-1    .code:n = { \keys_set:nn { enumext / print     } { print-1 = {#1} } } ,
4707      print-2    .code:n = { \keys_set:nn { enumext / print     } { print-2 = {#1} } } ,
4708      print-3    .code:n = { \keys_set:nn { enumext / print     } { print-3 = {#1} } } ,
4709      print-4    .code:n = { \keys_set:nn { enumext / print     } { print-4 = {#1} } } ,
4710      print-*    .code:n = { \keys_set:nn { enumext / print     } { print-* = {#1} } } ,
4711      unknown    .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
4712    }
```

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

```
4713 \seq_const_from_clist:Nn \c__enumext_all_families_seq
4714    {
4715      enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
4716      keyans*, print-1, print-2, print-3, print-4, print-*, print*,
4717    }
```

\setenumext   Now we define the user command \setenumext.

```
4718 \NewDocumentCommand \setenumext { O{enumext,1} +m }
4719    {
4720      \seq_clear:N \l__enumext_setkey_tmpa_seq
4721      \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
4722      \int_set:Nn \l__enumext_setkey_tmpa_int
4723        {
4724          \seq_count:N \l__enumext_setkey_tmpb_seq
4725        }
4726      \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
4727        {
4728          \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
4729          \seq_map_function:NN \l__enumext_setkey_tmpb_seq \__enumext_set_parse:n
4730          \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
4731            {
4732              \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
4733            }
4734        }
4735        {
4736          \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
4737        }
4738      \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
4739        { \seq_map_inline:Nn \c__enumext_all_families_seq }
4740        { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
4741        {
4742          \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
4743        }
4744    }
```

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

__enumext_set_parse:n   Internal functions used by the \setenumext command.
__enumext_set_error:nn

```
4745 \cs_new_protected:Npn \__enumext_set_parse:n #1
4746    {
4747      \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
4748      \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
4749        { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
4750      \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
4751        {
4752          \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
4753            { \tl_trim_spaces:n {#1} }
4754        }
4755        { \__enumext_set_error:nn {#1} { } }
4756    }
4757 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
4758    { \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.

```
4759 \prop_const_from_keyval:Nn \c__enumext_meta_paths_prop
4760   {
4761     {enumext,1} = level-1,
4762     {enumext,2} = level-2,
4763     {enumext,3} = level-3,
4764     {enumext,4} = level-4,
4765     {enumext*}  = enumext*
4766   }
```

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

```
4767 \NewDocumentCommand \setenumextmeta { s O{enumext,1} m +m }
4768   {
4769     \str_if_eq:eeTF { \tl_trim_spaces:n {#3} } { unknown }
4770       { \msg_error:nn { enumext } { prohibited-unknown } }
4771       {
4772         \bool_if:nTF {#1}
4773           {
4774             \int_step_inline:nn { 4 }
4775               { \__enumext_add_meta_key:nnn { enumext, ##1 } {#3} {#4} }
4776             \__enumext_add_meta_key:nnn { enumext* } {#3} {#4}
4777           }
4778           { \__enumext_add_meta_key:nnn {#2} {#3} {#4} }
4779       }
4780   }
```

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

```
4781 \cs_new_protected:Npn \__enumext_add_meta_key:nnn #1
4782   {
4783     \tl_set:Nn \l__enumext_meta_path_tl {#1}
4784     \tl_replace_all:Nnn \l__enumext_meta_path_tl { ~ } {}
4785     \prop_get:NVNTF
4786       \c__enumext_meta_paths_prop \l__enumext_meta_path_tl \l__enumext_meta_path_tl
4787       { \__enumext_def_meta_key:Vnn \l__enumext_meta_path_tl }
4788       {
4789         \msg_error:nnn { enumext } { unknown-set } {#1}
4790         \use_none:nn
4791       }
4792   }
4793 \cs_new_protected:Npn \__enumext_def_meta_key:nnn #1#2#3
4794   {
4795     \bool_lazy_or:nnTF
4796       { \keys_if_exist_p:nn { enumext / #1 } {#2} }
4797       { \keys_if_exist_p:nn { enumext / enumext* } {#2} }
4798       { \msg_error:nnn { enumext } { already-defined } {#2} }
4799       {
4800         \keys_define:nn { enumext / #1 }
4801           {
4802             #2 .meta:n = {#3},
4803             #2 .value_forbidden:n = true
4804           }
4805       }
4806   }
4807 \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 was given by Enrico Gregorio (@egreg) answer in Expand a .cs defined by key inside the function.

\foreachkayans

__enumext_parse_foreach_keys:nn
__enumext_parse_foreach_keys:n
__enumext_foreach_kayans: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.

```
4808 \keys_define:nn { enumext / foreach }
```

```
4809    {
4810      before  .tl_set:N  = \l__enumext_foreach_before_tl,
4811      before  .value_required:n = true,
4812      after   .tl_set:N  = \l__enumext_foreach_after_tl,
4813      after   .value_required:n = true,
4814      start   .int_set:N = \l__enumext_foreach_start_int,
4815      start   .value_required:n = true,
4816      stop    .int_set:N =  \l__enumext_foreach_stop_int,
4817      stop    .value_required:n = true,
4818      step    .int_set:N = \l__enumext_foreach_step_int,
4819      step    .value_required:n = true,
4820      wrapper .cs_set_protected:Np = \__enumext_foreach_wrapper:n #1,
4821      wrapper .value_required:n = true,
4822      sep     .tl_set:N  = \l__enumext_foreach_sep_tl,
4823      sep     .value_required:n = true,
4824      unknown .code:n    = { \__enumext_parse_foreach_keys:n {#1} }
4825    }
4826  \keys_precompile:nnN { enumext / foreach }
4827    {
4828      before={},after={},start=1,step=1,stop=0,wrapper=#1,sep=
4829    }
4830    \g__enumext_foreach_default_keys_tl
```

Functions for handling unknown ⟨*keys*⟩.

```
4831  \cs_new_protected:Npn \__enumext_parse_foreach_keys:nn #1#2
4832    {
4833      \tl_if_blank:nTF {#2}
4834        { \msg_error:nnn { enumext } { for-key-unknown } {#1} }
4835        { \msg_error:nnnn { enumext } { for-key-value-unknown } {#1} {#2} }
4836    }
4837  \cs_new_protected:Npn \__enumext_parse_foreach_keys:n #1
4838    {
4839      \exp_args:NV \__enumext_parse_foreach_keys:nn \l_keys_key_str {#1}
4840    }
```

We create the command.

```
4841  \NewDocumentCommand \foreachkayans { +O{} m }
4842    {
4843      \__enumext_foreach_kayans:nn {#1} {#2}
4844    }
```

Finally the internal functions __enumext_foreach_kayans:nn and __enumext_foreach_add_-
body:n will loop through the prop list and print the contents.

```
4845  \cs_new_protected:Npn \__enumext_foreach_kayans:nn #1 #2
4846    {
4847      \tl_use:N \g__enumext_foreach_default_keys_tl
4848      \keys_set:nn { enumext / foreach } {#1}
4849      \tl_set:Nn \l__enumext_foreach_name_prop_tl {#2}
4850      \seq_clear:N \l__enumext_foreach_print_seq
4851      \int_compare:nNnT { \l__enumext_foreach_stop_int } = { 0 }
4852        {
4853          \int_set:Nn \l__enumext_foreach_stop_int
4854            { \prop_count:c { g__enumext_#2_prop } }
4855        }
4856      \int_step_function:nnnN
4857        { \l__enumext_foreach_start_int }
4858        { \l__enumext_foreach_step_int }
4859        { \l__enumext_foreach_stop_int }
4860        \__enumext_foreach_add_body:n
4861        \seq_use:NV \l__enumext_foreach_print_seq \l__enumext_foreach_sep_tl
4862    }
4863  \cs_new_protected:Npn \__enumext_foreach_add_body:n #1
4864    {
4865      \seq_put_right:Ne \l__enumext_foreach_print_seq
4866        {
4867          \exp_not:V \l__enumext_foreach_before_tl
4868          \__enumext_foreach_wrapper:n
4869            {
4870              \prop_item:cn { g__enumext_ \l__enumext_foreach_name_prop_tl _prop }{#1}
4871            }
4872          \exp_not:V \l__enumext_foreach_after_tl
4873        }
4874    }
```

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

12.50 Messages

Message used by package-load for multicol and hyperref packages.

```
4875 \msg_new:nnn { enumext } { package-load }
4876   {
4877     The ~ '#1' ~ package ~ is ~ already ~ loaded.
4878   }
4879 \msg_new:nnn { enumext } { package-not-load }
4880   {
4881     The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
4882   }
4883 \msg_new:nnn { enumext } { package-load-foot }
4884   {
4885     The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
4886   }
```

Message used in the creation of counters by enumext package.

```
4887 \msg_new:nnn { enumext } { counters }
4888   {
4889     The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
4890     package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
4891   }
```

Message used by align and mark-pos keys.

```
4892 \msg_new:nnn { enumext } { unknown-choice }
4893   {
4894     The ~ value ~ '#3' ~ for ~ '#1' ~ key ~ is ~ invalid ~ use ~ ('#2').
4895   }
```

Message used by reserved anskey* environment by enumext package.

```
4896 \msg_new:nnnn { enumext } { anskey-env-error }
4897   {
4898     The ~ '#1' ~ environment ~is ~ reserved ~ by ~\\
4899     'enumext' ~ package, ~ It~ is~ already~ defined.
4900   }
4901   {
4902     The ~ anskey* ~ environment ~ is ~ defined ~ internally ~
4903     for ~ the ~ 'save-ans' ~ key.\\
4904   }
```

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

```
4905 \msg_new:nnn { enumext } { store-prop }
4906   {
4907     * ~ Package ~ enumext: ~ Creating ~
4908     \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
4909   }
4910 \msg_new:nnn { enumext } { store-seq }
4911   {
4912     * ~ Package ~ enumext: ~ Creating ~
4913     \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
4914   }
4915 \msg_new:nnn { enumext } { store-int }
4916   {
4917     * ~ Package ~ enumext: ~ Creating ~
4918     \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_context:.
4919   }
4920 \msg_new:nnn { enumext } { prop-seq-int-hook }
4921   {
4922     * ~ Package ~ enumext: ~ Elements ~ in ~
4923     \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
4924     * ~ Package ~ enumext: ~ Elements ~ in ~
4925     \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
4926     * ~ Package ~ enumext: ~ Value ~ off ~
4927     \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
4928   }
4929 \msg_new:nnn { enumext } { item-answer-hook }
4930   {
4931     * ~ Package ~ enumext: ~ Value ~ off ~
4932     \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\\
4933     * ~ Package ~ enumext: ~ Value ~ off ~
4934     \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\\
```

```
4935        * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
4936      }
```

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

```
4937  \msg_new:nnn { enumext } { invalid-key }
4938      {
4939        The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
4940      }
4941  \msg_new:nnn { enumext } { unknown-key-family }
4942      {
4943        Unknown~key~family~`\l_keys_key_str'~for~enumext.
4944      }
```

Messages used in length calculation.

```
4945  \msg_new:nnn { enumext } { width-negative }
4946      {
4947        Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\\
4948        The ~ key ~ '#1'~ accepts ~ values  ~ >= ~ 0pt.
4949      }
4950  \msg_new:nnn { enumext } { width-zero }
4951      {
4952        Invalid ~ '#1=#2' ~ \msg_line_context:.\\
4953        The ~ key ~ '#1'~ accepts ~ values  ~ > ~ 0pt.
4954      }
```

Messages used by show-length key in enumext.

```
4955  \msg_new:nnn { enumext } { list-lengths }
4956      {
4957        **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
4958        \__enumext_show_length:nnn { dim  } { labelsep     } {#1}
4959        \__enumext_show_length:nnn { dim  } { labelwidth   } {#1}
4960        \__enumext_show_length:nnn { dim  } { itemindent   } {#1}
4961        \__enumext_show_length:nnn { dim  } { leftmargin   } {#1}
4962        \__enumext_show_length:nnn { dim  } { rightmargin  } {#1}
4963        \__enumext_show_length:nnn { dim  } { listparindent } {#1}
4964        \__enumext_show_length:nnn { skip } { topsep       } {#1}
4965        \__enumext_show_length:nnn { skip } { parsep       } {#1}
4966        \__enumext_show_length:nnn { skip } { partopsep } {#1}
4967        \__enumext_show_length:nnn { skip } { itemsep    } {#1}
4968        **************************************************
4969      }
```

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

```
4970  \msg_new:nnn { enumext } { list-lengths-not-nested }
4971      {
4972        **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
4973        \__enumext_show_length:nnn { dim  } { labelsep     } {#1}
4974        \__enumext_show_length:nnn { dim  } { labelwidth   } {#1}
4975        \__enumext_show_length:nnn { dim  } { itemindent   } {#1}
4976        \__enumext_show_length:nnn { dim  } { leftmargin   } {#1}
4977        \__enumext_show_length:nnn { dim  } { rightmargin  } {#1}
4978        \__enumext_show_length:nnn { dim  } { listparindent } {#1}
4979        \__enumext_show_length:nnn { skip } { topsep       } {#1}
4980        \__enumext_show_length:nnn { skip } { parsep       } {#1}
4981        \__enumext_show_length:nnn { skip } { partopsep } {#1}
4982        \__enumext_show_length:nnn { skip } { itemsep    } {#1}
4983        **************************************************
4984      }
```

Messages used by ref key.

```
4985  \msg_new:nnn { enumext } { key-ref-empty }
4986      {
4987        Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4988      }
```

Messages used by save-ans key.

```
4989  \msg_new:nnn { enumext } { save-ans-empty }
4990      {
4991        Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4992      }
4993  \msg_new:nnn { enumext } { save-ans-log }
4994      {
4995        * ~ Package ~ enumext: ~ Start ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
4996      }
```

```
4997 \msg_new:nnn { enumext } { save-ans-log-hook }
4998   {
4999     * ~ Package ~ enumext: ~ Stop ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5000   }
5001 \msg_new:nnn { enumext } { save-ans-hook }
5002   {
5003     Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
5004   }
```

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

```
5005 \msg_new:nnn { enumext } { need-save-ans }
5006   {
5007     Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
5008   }
5009 \msg_new:nnn { enumext } { items-same-answer }
5010   {
5011     ****************************************\\
5012     * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~
5013     for ~ \c_left_brace_str #2 \c_right_brace_str\\
5014     * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~
5015     'OK', ~ all ~ items ~ with ~ answer.\\
5016     ****************************************
5017   }
5018 \msg_new:nnn { enumext } { item-greater-answer }
5019   {
5020     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
5021     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
5022     Items ~ > ~ Answers.
5023   }
5024 \msg_new:nnn { enumext } { item-less-answer }
5025   {
5026     Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
5027     started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
5028     Items ~ < ~ Answers.
5029   }
```

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

```
5030 \msg_new:nnn { enumext } { missing-starred }
5031   {
5032     Missing ~ '\c_backslash_str #1*' ~ #2.
5033   }
5034 \msg_new:nnn { enumext } { many-starred }
5035   {
5036     Many ~ '\c_backslash_str #1*' ~ #2.
5037   }
```

Messages used by \printkeyans* command.

```
5038 \msg_new:nnn { enumext } { print-starred }
5039   {
5040     \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
5041     #2 ~ environment ~  \msg_line_context:.
5042   }
```

Message for the nesting depth of the environment enumext.

```
5043 \msg_new:nnn { enumext } { list-too-deep }
5044   {
5045     Too ~ deep ~ nesting  ~ for ~ 'enumext' ~ \msg_line_context:.~ \\
5046     The ~ maximum  ~ level  ~ of  ~ nesting  ~ is ~ 4.
5047   }
```

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

```
5048 \msg_new:nnn { enumext } { anskey-unnumber-item }
5049   {
5050     Can't ~ store ~ with ~ a ~ unnumbered ~ \c_backslash_str item ~ \msg_line_context:.
5051   }
5052 \msg_new:nnn { enumext } { anskey-already-stored }
5053   {
5054     Content ~ already ~ stored ~ for ~ this ~ \c_backslash_str item ~ \msg_line_context:.
5055   }
5056 \msg_new:nnn { enumext } { anskey-empty-arg }
5057   {
5058     Can't ~ store ~ empty ~ content ~ \msg_line_context:.
5059   }
```

```
5060 \msg_new:nnn { enumext } { anskey-wrong-place }
5061   {
5062     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
5063     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5064   }
5065 \msg_new:nnn { enumext } { anskey-nested }
5066   {
5067     The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
5068   }
5069 \msg_new:nnn { enumext } { anskey-math-mode }
5070   {
5071     #1 ~ can't ~ work ~ in ~ math ~ mode ~ \msg_line_context:.
5072   }
5073 \msg_new:nnn { enumext } { anskey-env-wrong }
5074   {
5075     The ~ environment ~ anskey* ~ cannot ~ use ~ in ~ '#1' ~ \msg_line_context:.
5076   }
5077 \msg_new:nnn { enumext } { anspic-wrong-place }
5078   {
5079     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
5080     '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5081   }
5082 \msg_new:nnn { enumext } { command-wrong-place }
5083   {
5084     Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
5085     '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
5086   }
5087 \msg_new:nnnn { enumext } { anskey-env-key-unknown }
5088   {
5089     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5090     'anskey*' ~ and ~ is ~ being ~ ignored.
5091   }
5092   {
5093     The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5094     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5095   }
5096 \msg_new:nnnn { enumext } { anskey-env-key-value-unknown }
5097   {
5098     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5099     'anskey*' ~ and ~ is ~ being ~ ignored.
5100   }
5101   {
5102     The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5103     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5104   }
5105 \msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
5106   { The ~ key ~'#1'~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.}
5107   {
5108     The ~ command ~'\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5109     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5110   }
5111 \msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
5112   { The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignor
5113   {
5114     The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5115     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5116   }
```

Messages used by `keyans`, `keyans*` and `keyanspic` environment.

```
5117 \msg_new:nnn { enumext } { keyans-nested }
5118   {
5119     The ~ environment ~ 'keyans' ~ can't ~ be  ~ nested  ~ \msg_line_context:.
5120   }
5121 \msg_new:nnn { enumext } { keyans-wrong-level }
5122   {
5123     Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:.~ \\
5124     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5125   }
5126 \msg_new:nnn { enumext } { wrong-place }
5127   {
5128     Wrong ~ place ~ for ~ '#1' ~ environment ~\msg_line_context:.~ \\
5129     '#1' ~ is ~ only ~ found ~ with ~ '#2' ~  in  ~  'enumext.
```

```
5130        }
5131  \msg_new:nnn { enumext } { keyanspic-nested }
5132    {
5133      The ~ environment ~ 'keyanspic' ~ can't ~ be  ~ nested~ \msg_line_context:.~.
5134    }
5135  \msg_new:nnn { enumext } { keyanspic-wrong-level }
5136    {
5137      Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:.~ \\
5138      The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5139    }
5140  \msg_new:nnn { enumext } { keyanspic-item-cmd }
5141    {
5142      Can't ~ use  ~ \c_backslash_str item ~ in ~ keyanspic ~ \msg_line_context:.
5143    }
5144  \msg_new:nnnn { enumext } { keyans-unknown-key }
5145    {
5146      The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5147      '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5148    }
5149    {
5150      The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5151      ~ have ~ a ~ key ~ called ~'#1'.\\
5152      Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5153    }
5154  \msg_new:nnnn { enumext } { keyans-unknown-key-value }
5155    {
5156      The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5157      '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5158    }
5159    {
5160      The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5161      ~ have ~ a ~ key ~ called ~'#1'.\\
5162      Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5163    }
```

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

```
5164  \msg_new:nnnn { enumext } { starred-unknown-key }
5165    {
5166      The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5167      '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5168    }
5169    {
5170      The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5171      ~ have ~ a ~ key ~ called ~'#1'.\\
5172      Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5173    }
5174  \msg_new:nnnn { enumext } { starred-unknown-key-value }
5175    {
5176      The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5177      '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5178    }
5179    {
5180      The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5181      ~ have ~ a ~ key ~ called ~'#1'.\\
5182      Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5183    }
```

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

```
5184  \msg_new:nnnn { enumext } { standar-unknown-key }
5185    {
5186      The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_tl
5187      ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5188    }
5189    {
5190      The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5191      ~ have ~ a ~ key ~ called ~'#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5192      Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5193    }
5194  \msg_new:nnnn { enumext } { standar-unknown-key-value }
5195    {
5196      The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_t
5197      ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
```

```
5198        }
5199      {
5200        The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5201        ~ have ~ a ~ key ~ called ~'#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5202        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5203      }
```

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

```
5204 \msg_new:nnnn { enumext } { for-key-unknown }
5205      { The~key~'#1'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored.}
5206      {
5207        The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5208        Check~that~you~have~spelled~the~key~name~correctly.
5209      }
5210 \msg_new:nnnn { enumext } { for-key-value-unknown }
5211      { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored. }
5212      {
5213        The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5214        Check~that~you~have~spelled~the~key~name~correctly.
5215      }
```

Messages used by \getkeyans command.

```
5216 \msg_new:nnn { enumext } { undefined-storage-anskey }
5217      {
5218        Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
5219      }
```

Messages used by \miniright command.

```
5220 \msg_new:nnn { enumext } { missing-miniright }
5221      {
5222        Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\\
5223        The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
5224      }
5225 \msg_new:nnn { enumext } { wrong-miniright-place }
5226      {
5227        Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
5228        Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
5229      }
5230 \msg_new:nnn { enumext } { wrong-miniright-use }
5231      {
5232        Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
5233        '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
5234      }
5235 \msg_new:nnn { enumext } { wrong-miniright-starred }
5236      {
5237        Can't ~ use  ~ \c_backslash_str miniright ~ in ~ starred ~ environments ~ \msg_line_context:.
5238      }
5239 \msg_new:nnn { enumext } { many-miniright-used }
5240      {
5241        Can't ~ use  ~ \c_backslash_str miniright ~ more ~ than ~ once ~  \msg_line_context:.
5242      }
```

Messages used by \setenumextmeta command.

```
5243 \msg_new:nnn { enumext } { unknown-set }
5244      {
5245        Argument ~ [#1] ~ is ~ unknown ~ by ~  \c_backslash_str setenumextmeta ~ \msg_line_context:.
5246      }
5247 \msg_new:nnn { enumext } { already-defined }
5248      {
5249        The ~ key ~ '#1' ~ is ~ already ~ defined ~ \msg_line_context:.
5250      }
5251 \msg_new:nnn { enumext } { prohibited-unknown }
5252      {
5253        The ~ name ~ 'unknown' ~ can't ~ be ~ chosen~ for ~ a ~ meta ~ key ~ \msg_line_context:.
5254      }
```

Messages used by enumext* and keyans* environments.

```
5255 \msg_new:nnn { enumext } { nested }
5256      {
5257        The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ \msg_line_con
5258      }
5259 \msg_new:nnn { enumext } { nested-horizontal }
5260      {
```

```
5261         The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ in ~ '#1' ~ '
5262      }
5263  \msg_new:nnn { enumext } { item-joined }
5264      {
5265         Items ~ joined ~ (#1) ~ > ~ #2  ~ columns ~\msg_line_context:.
5266      }
5267  \msg_new:nnn { enumext } { item-joined-columns }
5268      {
5269         Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.
5270      }
```

12.51   Finish package

Finish package implementation.

```
5271  \file_input_stop:
5272  ⟨/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.