enumext

ENUMERATE EXERCISE SHEETS

V1.0   2024-06-13[*]

©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

---

[*]This file describes a documentation for v1.0, last revised 2024-06-13.
[†]E-mail: «pablgonz@educarchile.cl».

License and Requirements

Permission is granted to copy, distribute and/or modify this software under the terms of the LaTeX Project Public License (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 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.
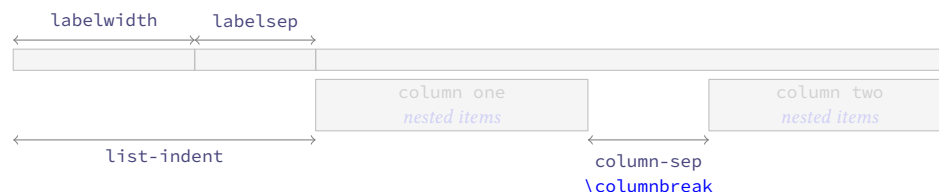

Figure 4: Representation of the two column output for a nested level in `enumext` environment.

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 §4.5).

💣 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.

1.3.3 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 §4.6).

1.3.4 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 (§4.1), 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` (§5.1.1) when the key `save-ans` (§5.1) is active.

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

1.3.5 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, but it can be nested within enumext and can contain it nested within it.

- Each *"item"* in the environment is placed within a minipage environment whose *width* is stored in the dimension \itemwidth that includes labelwith, labelsep plus 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 value set by the labelsep key and can be ⟨offset⟩ using 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
  \item[Z] The ninth
  \item The tenth
\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   Z The ninth

8. The tenth

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 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.

4.1 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⟩]`.

4.2 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`.

4.2.1 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 `keyans`, `keyans*` and `enumext*` environments the default value is `2.0pt plus 1.0pt minus 1.0pt`.

💣 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"*.

4.2.2 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.

4.3 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 `\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*⟩`}\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`: `\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.

4.4 Keys for start, series and resume

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

Sets the *start value* of the numbering on the current level. Internally ⟨*string*⟩ is 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 *"first level"* of `enumext` and `enumext*` and are ignored if set when nested inside 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 the `start` key.

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 the start key.

💣 For security reasons the series key will never save in {⟨*series name*⟩} the keys series, resume, resume*, save-ans, save-key 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*.

4.5 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.

4.6 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.

4.6.1 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 §1.3.5 for full support).

4.6.2 The key mini-right

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

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

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

mini-right* = {⟨*code for drawing or tabular*⟩} 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*.

5 The storage system

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

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

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

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

5.1 Keys for storage system

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

`save-ans` = {⟨*store name*⟩}   default: *not set*

Sets the *name* of the ⟨*sequence*⟩ and ⟨*prop list*⟩ in which the contents will be *"stored"* by `\anskey` and `anskey*` in `enumext` and `enumext*` environments, `\item*` in `keyans` and `keyans*` environments and `\anspic*` in `keyanspic` environment. If the ⟨*sequence*⟩ or ⟨*prop list*⟩ does not exist, it will be created globally and will not be overwritten if the key is used again.

`save-key` = {⟨*key list*⟩}   default: *not set*

This key *overrides* the default *"stored keys"* of the optional arguments of the inner levels or nested environment that will be passed to the ⟨*sequence*⟩. The ⟨*key list*⟩ passed to this key ignores any ⟨*keys*⟩ in the *"stored system"* and must be passed between braces. For example, if we execute at a second level:

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

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

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

Sets the *text symbol* that will separate the current ⟨*label*⟩ to the *optional argument* passed to the `\item*` and `\anspic*` in the `keyans`, `keyans*` and `keyanspic` environments and storing them in the ⟨*store name*⟩ defined by the `save-ans` key. The {⟨*text symbol*⟩} must always be passed between braces, whitespace '␣' is preserved within the braces and only affects the *"stored content"* and not what is displayed when using the `show-ans` or `show-pos` keys.

5.1.1 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.

5.1.2 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.

5.1.3 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*⟩}.

5.2 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.

5.2.1 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 |

5.3 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.

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

💣 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 |

5.4 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*)`.

5.4.1 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

5.5 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.

5.5.1 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}
```
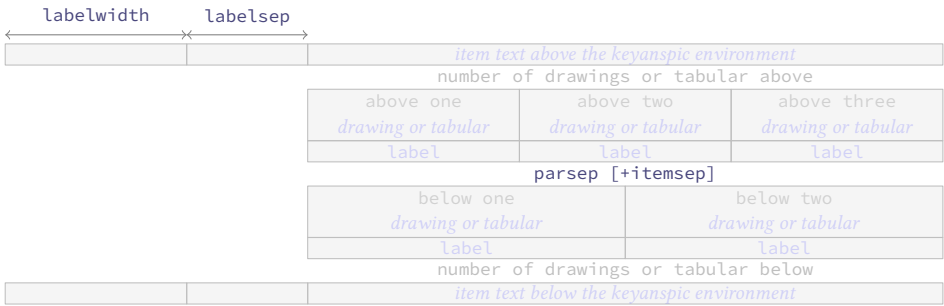
1. Question with images.

A)

B)

C)

D)

* E)[note]

5.6 Printing stored content

5.6.1 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.

5.6.2 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   *

6 Full examples

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

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

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

Example 1

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

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

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

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

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

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

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

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

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

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

Example 2

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

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

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

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

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

---

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

D | $1\,Å = 1 \times 10^3$ 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: $1\,Å = 1 \times 10^{-10}$ m) e il fermi o femtometro ($1\,$fm $= 1 \times 10^{-15}$ m). Qual è la relazione tra queste due unità di misura?

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

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

Example 3

A *"simple multiple choice"* test 📄.

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

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

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

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

   Ⓐ value
   Ⓑ value
   Ⓒ value
   Ⓓ value
   Ⓔ value

4. Question with image and label below:

Ⓐ

Ⓑ

Ⓒ

Ⓓ

Ⓔ

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

Test keys

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

Example 4

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

🦆 Factor $x^2 - 2x + 1$

🦆 Factor $3x + 3y + 3z$

The following questions need to be cuaqtified :)

3 True False
(a) $\alpha > \delta$
(b) LATEX2e is cool?

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

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

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

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

Example 5

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

1

Which choice best describes what happens in the passage?

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

2

Which choice best describes what happens in the passage?

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

3

Which choice best describes what happens in the passage?

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

4

Which choice best describes what happens in the passage?

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

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

7 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 typessetting 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.

8 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.

9 Change history

v1.0 2024-06-13 – First public release.

10 Index of Documentation

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

11 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.

11.1 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.

11.2 Initial set up

Start the DocStrip guards.

```
1  ⟨*package⟩
```

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

```
2  ⟨@@=enumext⟩
```

11.3 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    {2024-06-13}
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   }
```

11.4 Definition of variables

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

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

Integer variables will control the nesting levels of the environments and `\anskey` command.

```
28 \int_new:N  \l__enumext_level_int
29 \int_new:N  \l__enumext_level_h_int
30 \int_new:N  \l__enumext_anskey_level_int
31 \int_new:N  \l__enumext_keyans_level_int
32 \int_new:N  \l__enumext_keyans_level_h_int
33 \int_new:N  \l__enumext_keyans_pic_level_int
```

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

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

Internal variables used by functions `\__enumext_is_not_nested:`, `\__enumext_is_on_first_-level:` and `\__enumext_keyans_start_line:` (§11.6.1).

```
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
```

(*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` (§11.10) and then modified by the function `\__enumext_label_style:Nnn` used by label key (§11.13).

```
44 \cs_set_protected:Npn \__enumext_tmp:n #1
45   {
46     \tl_new:c { l__enumext_counter_#1_tl }
47   }
48 \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 (§11.13).

```
49 \tl_const:Nn \c__enumext_counter_style_tl
50   { { arabic } { roman } { Roman } { alph } { Alph } }
51 \tl_new:N \l__enumext_ref_key_arg_tl
52 \tl_new:N \l__enumext_ref_the_count_tl
53 \cs_set_protected:Npn \__enumext_tmp:n #1
54   {
55     \tl_new:c  { l__enumext_renew_the_count_#1_tl }
56     \tl_new:c  { l__enumext_the_counter_#1_tl }
57     \tl_set:ce { l__enumext_the_counter_#1_tl } { \exp_not:c { theenumX#1 } }
58   }
59 \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`
`\g__enumext_item_symbol_tl`

Internal variables used by resume, resume* and series keys (§11.24).

```
60 \int_new:N  \g__enumext_resume_int
61 \int_new:N  \g__enumext_resume_vii_int
62 \tl_new:N    \l__enumext_resume_name_tl
63 \bool_new:N \l__enumext_resume_active_bool
64 \tl_new:N    \g__enumext_standar_series_tl
65 \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 (§11.14) and label (§11.12) keys.

```
66 \dim_new:N \l__enumext_current_widest_dim
67 \tl_new:N  \g__enumext_counter_styles_tl
68 \tl_new:N  \g__enumext_widest_label_tl
69 \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 (§11.17). The variables `\l__enumext_leftmargin_X_dim` and `\l__enumext_itemindent_X_dim` are used and set by the function `\__enumext_calc_hspace:NNNNNNNNNNNNNN` (§11.36.1).

```
70 \cs_set_protected:Npn \__enumext_tmp:n #1
71   {
72     \bool_new:c { l__enumext_leftmargin_tmp_#1_bool }
73     \dim_new:c  { l__enumext_leftmargin_tmp_#1_dim }
74     \dim_new:c  { l__enumext_leftmargin_#1_dim     }
75     \dim_new:c  { l__enumext_itemindent_#1_dim     }
76   }
77 \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 §11.21).

```
78 \cs_set_protected:Npn \__enumext_tmp:n #1
79   {
80     \skip_new:c  { l__enumext_multicols_above_#1_skip }
81     \skip_new:c  { l__enumext_multicols_below_#1_skip }
82   }
83 \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 (§11.22.4) and the keys mini-right, mini-right*, mini-env and mini-sep (§11.20, §11.22).

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

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

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

The integer variable `\l__enumext_start_X_int` are used by the start key (§11.14), the token list `\l__enumext_fake_item_indent_X_tl` is used by itemindent key (§11.17.1), the variables `\l__enumext_label_fill_left_X_tl` and `\l__enumext_label_fill_left_X_tl` are used by the align key (§11.12). 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 (§11.19).

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

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

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

The variable `\l__enumext_store_active_bool` setting by save-ans key (§11.25.1) 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 (§11.29) 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* (§11.30).

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* (§11.35) and \anspic* (§11.39.1) 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.

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

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

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

Internal variables used by the command \setenumext (§11.46).

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

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

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

Internal variables used by command \printkeyans (§11.45), show-pos key (§11.26), item-sym* key (§11.34), save-key key (§11.26.2) and *"storage level system"*.

```
122 \tl_new:N  \l__enumext_print_keyans_starred_tl
123 \str_new:N \l__enumext_mark_position_str
124 \tl_new:N  \g__enumext_item_symbol_tl
125 \cs_set_protected:Npn \__enumext_tmp:n #1
126   {
127     \tl_new:c   { l__enumext_print_keyans_#1_tl         }
128     \tl_new:c   { l__enumext_store_save_key_#1_tl        }
129     \bool_new:c { l__enumext_store_save_key_#1_bool      }
130     \bool_new:c { l__enumext_store_upper_level_#1_bool }
131   }
132 \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {#1} }
```

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

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

Internal variables used by keyanspic environment (§11.39.2).

```
133 \seq_new:N  \l__enumext_keyans_pic_body_seq
134 \dim_new:N  \l__enumext_keyans_pic_width_dim
135 \int_new:N  \l__enumext_keyans_pic_above_int
136 \int_new:N  \l__enumext_keyans_pic_below_int
137 \skip_new:N \l__enumext_keyans_pic_above_skip
```

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

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

Internal variables used by *"internal check answer"* mechanism (§11.25.3) 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.

```
138 \bool_new:N \l__enumext_check_answers_bool
139 \bool_new:N \g__enumext_check_ans_key_bool
140 \tl_new:N   \l__enumext_check_start_line_env_tl
141 \int_new:N  \g__enumext_check_starred_cmd_int
142 \int_new:N  \g__enumext_item_anskey_int
```

```
143 \int_new:N  \g__enumext_item_number_int
144 \bool_new:N \l__enumext_item_number_bool
145 \int_new:N  \g__enumext_item_answer_diff_int
```

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

\l__enumext_hyperref_bool
\l__enumext_footnotes_key_bool

The boolean variable \l__enumext_hyperref_bool will determine if the hyperref package is present or load in memory (§11.9). The boolean variable \l__enumext_footnotes_key_bool determine if hyperref is load with key hyperfootnotes=true.

```
146 \bool_new:N \l__enumext_hyperref_bool
147 \bool_new:N \l__enumext_footnotes_key_bool
```

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

\l__enumext_newlabel_arg_one_tl
\l__enumext_newlabel_arg_two_tl
\l__enumext_write_aux_file_tl
\l__enumext_label_copy_X_tl

Internal variables used by save-ref key (§11.26). 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 (§11.9) and the variable \l__enumext_write_aux_file_tl will be in charge of executing the writing code in the .aux file.

```
148 \tl_new:N \l__enumext_newlabel_arg_one_tl
149 \tl_new:N \l__enumext_newlabel_arg_two_tl
150 \tl_new:N \l__enumext_write_aux_file_tl
151 \cs_set_protected:Npn \__enumext_tmp:n #1
152   {
153     \tl_new:c { l__enumext_label_copy_#1_tl }
154   }
155 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }
```

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

\g__enumext_footnote_int
\g__enumext_footnote_arg_seq
\g__enumext_footnote_int_seq

Internal variables used for redefinition of \footnote (§11.41).

```
156 \int_new:N \g__enumext_footnote_int
157 \seq_new:N \g__enumext_footnote_arg_seq
158 \seq_new:N \g__enumext_footnote_int_seq
```

(*End of definition for* \g__enumext_footnote_int*,* \g__enumext_footnote_arg_seq*, and* \g__enumext_footnote_int_-seq*.*)

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

Internal variables used by enumext* and keyans* environments.

```
159 \cs_set_protected:Npn \__enumext_tmp:n #1
160   {
161     \bool_new:c { l__enumext_item_starred_#1_bool   }
162     \int_new:c  { l__enumext_item_column_pos_#1_int  }
163     \int_new:c  { g__enumext_item_count_all_#1_int   }
164     \int_new:c  { l__enumext_joined_item_#1_int      }
165     \int_new:c  { l__enumext_joined_item_aux_#1_int  }
166     \int_new:c  { l__enumext_tmpa_#1_int             }
167     \dim_new:c  { l__enumext_tmpa_#1_dim             }
168     \box_new:c  { l__enumext_item_text_#1_box        }
169     \dim_new:c  { l__enumext_joined_width_#1_dim     }
170     \dim_new:c  { l__enumext_item_width_#1_dim       }
171     \tl_new:c   { g__enumext_item_symbol_aux_#1_tl   }
172     \str_new:c  { l__enumext_align_label_#1_str      }
173     \bool_new:c { g__enumext_minipage_active_#1_bool }
174     \box_new:c  { l__enumext_miniright_code_#1_box   }
175     \bool_new:c { g__enumext_minipage_center_#1_bool }
176     \dim_new:c  { g__enumext_minipage_right_#1_dim   }
177     \skip_new:c { g__enumext_minipage_right_#1_skip  }
178   }
179 \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.

```
180 \clist_const:Nn \c__enumext_all_envs_clist
181   {
182     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
183     {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
184   }
```

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

11.5 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.

```
185 \dim_zero_new:N \itemwidth
```

11.6 Some utility functions

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

```
186 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
187   {
188     \hook_gput_code:nnn {begindocument} {enumext} { #1 }
189   }
```

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

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

```
190 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
191   {
192     \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
193   }
194 \cs_new_protected:Npn \__enumext_before_env:nn #1 #2
195   {
196     \hook_gput_code:nnn {env/#1/before} {enumext} {#2}
197   }
```

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

__enumext_level: Function for check current level in enumext.

```
198 \cs_new:Nn \__enumext_level:
199   {
200     \int_to_roman:n { \l__enumext_level_int }
201   }
```

(*End of definition for* __enumext_level:.)

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

```
202 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
203   {
204     \regex_match:nnTF { ^[\+\-]?[\d]+$ } {#1} % $
205       { \prg_return_true: }
206       { \prg_return_false: }
207   }
```

(*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.

```
208 \cs_new_protected:Nn \__enumext_regex_counter_style:
209   {
210     \tl_map_inline:Nn \c__enumext_counter_style_tl
211       {
212         \regex_replace_once:nnN { \c{##1}\* }
213           { \c{##1}\cB{\u{l__enumext_ref_the_count_tl}\cE} } \l__enumext_ref_key_arg_tl
214       }
215   }
```

(*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.

```
216  \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
217    {
218      * ~ #2
219      \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
220        = ~ \use:c { #1_use:c } { l__enumext_#2_#3_#1 } \\
221    }
```

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

11.6.1   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.

```
222  \cs_new_protected:Nn \__enumext_is_not_nested:
223    {
224      \str_case:en { \@currenvir }
225        {
226          {enumext}
227            {
228              \bool_lazy_and:nnT
229                { \bool_not_p:n { \g__enumext_standar_bool } }
230                { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
231                {
232                  \bool_gset_true:N \g__enumext_standar_bool
233                }
234            }
235          {enumext*}
236            {
237              \bool_lazy_and:nnT
238                { \bool_not_p:n { \g__enumext_starred_bool } }
239                { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
240                {
241                  \bool_gset_true:N \g__enumext_starred_bool
242                }
243            }
244        }
245    }
```

The function __enumext_is_on_first_level: will set the variables \l__enumext_standar_-first_bool (§11.25.1), \l__enumext_starred_first_bool (§11.25.1) and \l__enumext_anskey_-env_bool (§11.30) 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.

```
246  \cs_new_protected:Nn \__enumext_is_on_first_level:
247    {
248      \bool_lazy_all:nT
249        {
250          { \bool_if_p:N \g__enumext_standar_bool }
251          { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
252          { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
253        }
254        {
255          \bool_set_true:N \l__enumext_standar_first_bool
256          \bool_set_true:N \l__enumext_anskey_env_bool
257          \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
258          \tl_gset:Ne \g__enumext_start_line_tl
259            {
260              on ~ line ~ \exp_not:V \inputlineno
261            }
262        }
263      \bool_lazy_all:nT
264        {
265          { \bool_if_p:N \g__enumext_starred_bool }
266          { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
267          { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
268        }
269        {
270          \bool_set_true:N \l__enumext_starred_first_bool
```

```
271        \bool_set_true:N \l__enumext_anskey_env_bool
272        \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
273        \tl_gset:Ne \g__enumext_start_line_tl
274          {
275            on ~ line ~ \exp_not:V \inputlineno
276          }
277      }
278    }
```

(*End of definition for* __enumext_is_not_nested: *and* __enumext_is_on_first_level:.)

__enumext_keyans_start_line:

The function __enumext_keyans_start_line: will save the start line number of the environments keyans, keyans* and keyanspic in the variable \l__enumext _check_start_line_env_tl to use in the __enumext_check_starred_cmd:n function.

```
279  \cs_new_protected:Nn \__enumext_keyans_start_line:
280    {
281      \str_case:en { \@currenvir }
282        {
283          {keyans}
284            {
285              \tl_set:Ne \l__enumext_check_start_line_env_tl
286                {
287                  in ~ 'keyans' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
288                }
289            }
290          {keyans*}
291            {
292              \tl_set:Ne \l__enumext_check_start_line_env_tl
293                {
294                  in ~ 'keyans*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
295                }
296            }
297          {keyanspic}
298            {
299              \tl_set:Ne \l__enumext_check_start_line_env_tl
300                {
301                  in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
302                }
303            }
304        }
305    }
```

(*End of definition for* __enumext_keyans_start_line:.)

11.6.2 Utilities for log and terminal

__enumext_reset_global_vars:
__enumext_reset_global_int:
__enumext_reset_global_bool:
__enumext_reset_global_tl:

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

```
306  \cs_new_protected:Nn \__enumext_reset_global_vars:
307    {
308      \__enumext_reset_global_int:
309      \__enumext_reset_global_bool:
310      \__enumext_reset_global_tl:
311    }
312  \cs_new_protected:Nn \__enumext_reset_global_int:
313    {
314      \int_gzero:N \g__enumext_item_number_int
315      \int_gzero:N \g__enumext_item_anskey_int
316      \int_gzero:N \g__enumext_item_answer_diff_int
317    }
318  \cs_new_protected:Nn \__enumext_reset_global_bool:
319    {
320      \bool_gset_false:N \g__enumext_check_ans_key_bool
321      \bool_gset_false:N \g__enumext_standar_bool
322      \bool_gset_false:N \g__enumext_starred_bool
323    }
324  \cs_new_protected:Nn \__enumext_reset_global_tl:
325    {
326      \tl_gclear:N \g__enumext_store_name_tl
327      \tl_gclear:N \g__enumext_start_line_tl
328      \tl_gclear:N \g__enumext_envir_name_tl
329    }
```

(*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.

```
330  \cs_new_protected:Nn \__enumext_log_global_vars:
331    {
332      \msg_log:nneeee { enumext } { prop-seq-int-hook }
333        { \g__enumext_store_name_tl }
334        { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
335        { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
336        { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
337    }
```

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.

```
338  \cs_new_protected:Nn \__enumext_log_answer_vars:
339    {
340      \msg_log:nneee { enumext } { item-answer-hook }
341        { \int_use:N \g__enumext_item_number_int }
342        { \int_use:N \g__enumext_item_anskey_int }
343        { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int} }
344    }
```

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

11.7 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.

```
345  \__enumext_at_begin_document:n
346    {
347      \cs_new_eq:NN \__enumext_start_list:nn \list
348      \cs_new_eq:NN \__enumext_stop_list: \endlist
349      \cs_new_eq:NN \__enumext_item_std:w \item
350    }
```

(*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.

```
351  \__enumext_at_begin_document:n
352    {
353      \cs_new_eq:NN \__enumext_minipage:w \minipage
354      \cs_new_eq:NN \__enumext_endminipage: \endminipage
355    }
```

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

11.8 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 `\vspace{0pt}` to maintain alignment on *"top"*. 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 (§11.37) and `\__enumext_safe_exec_vii:` in the `enumext*` environment definition (§11.42)

```
356  \cs_new_protected:Nn \__enumext_internal_mini_page:
357    {
358      \int_compare:nNnT { \l__enumext_level_int } = { 0 }
359        {
360          \DeclareDocumentEnvironment{__enumext_mini_env*}{ m }
361            {
362              \__enumext_minipage:w [ t ] { ##1 }
363                \legacy_if_gset_false:n { @minipage }
364                \vspace { 0pt }
365            }
366            { \__enumext_endminipage: }
367        }
368    }
```

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

11.9 Compatibility with hyperref and footnotehyper

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

```
369  \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
370  \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".

```
371  \cs_new_protected:Nn \__enumext_after_hyperref:
372    {
373      \IfPackageLoadedTF { hyperref }
374        {
375          \msg_info:nnn { enumext } { package-load } { hyperref }
376          \bool_set_true:N \l__enumext_hyperref_bool
377          \IfHyperBoolean{hyperfootnotes}
378            {
379              \typeout{hyperfootnotes=true}
380              \bool_set_true:N \l__enumext_footnotes_key_bool
381            }
382            { \typeout{hyperfootnotes=false} }
383        }
384        {  }
```

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`.

```
385      \bool_if:NT \l__enumext_footnotes_key_bool
386        {
387          \IfPackageLoadedTF { footnotehyper }
388            {
389              \msg_info:nnn { enumext } { package-load } { footnotehyper }
390            }
391            {
392              \typeout{No ~ footnotehyper ~ load}
393              \typeout{Load ~ and  ~ use  ~ \string\makesavenoteenv{enumext*}}
394              \bool_set_false:N \l__enumext_footnotes_key_bool
395            }
396        }
```

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.

```
397      \bool_if:NTF \l__enumext_hyperref_bool
398        {
399          \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
400          \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
401        }
402        {
403          \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
404          \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
405        }
406      }
```

(*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.

```
407  \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
408    {
409      \protected@write \@auxout { }
410        {
411          \token_to_str:N \newlabel {#1}
412            {
413              {#2}
414              \bool_if:NT \l__enumext_hyperref_bool
415                { { \thepage } {#2} {#1} }
416              { }
417            }
418        }
419      \__enumext_hypertarget:nn {#1} { }
420      \__enumext_phantomsection:
421    }
```

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

11.10   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.

```
422  \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
423    {
424      \cs_if_exist:cTF { c@ #2 }
425        { \msg_fatal:nnn { enumext } { counters }{ #2 } }
426        {
427          \tl_set:Nn #1 { #2 }
428          \newcounter { #2 }
429        }
430    }
```

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

enumXi
enumXii
enumXiii
enumXiv
enumXv
enumXvi
enumXvii
enumXviii

The counters created here are enumXi, enumXii, enumXiii and enumXiv for enumext environment, enumXv for keyans environment, enumXvi for keyanspic environment, enumXvii for enumext* and enumXviii for the keyans* environments.

```
431  \__enumext_define_counters:Nn \l__enumext_counter_i_tl    { enumXi     }
432  \__enumext_define_counters:Nn \l__enumext_counter_ii_tl   { enumXii    }
433  \__enumext_define_counters:Nn \l__enumext_counter_iii_tl  { enumXiii   }
434  \__enumext_define_counters:Nn \l__enumext_counter_iv_tl   { enumXiv    }
435  \__enumext_define_counters:Nn \l__enumext_counter_v_tl    { enumXv     }
436  \__enumext_define_counters:Nn \l__enumext_counter_vi_tl   { enumXvi    }
437  \__enumext_define_counters:Nn \l__enumext_counter_vii_tl  { enumXvii   }
438  \__enumext_define_counters:Nn \l__enumext_counter_viii_tl { enumXviii  }
```

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

11.11 Definition of labels

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

`\__enumext_register_counter_style:Nn`

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

```
439  \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
440    {
441      \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
442      \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
443    }
444  \__enumext_register_counter_style:Nn \arabic { 0 }
445  \__enumext_register_counter_style:Nn \Alph   { M }
446  \__enumext_register_counter_style:Nn \alph   { m }
447  \__enumext_register_counter_style:Nn \Roman  { VIII }
448  \__enumext_register_counter_style:Nn \roman  { viii }
```

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

`\__enumext_label_width_by_box:Nn`
`\__enumext_label_width_by_box:cv`

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

```
449  \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
450    {
451      \hbox_set:Nn \l__enumext_label_width_by_box {#2}
452      \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
453    }
454  \cs_generate_variant:Nn \__enumext_label_width_by_box:Nn { cv }
```

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

`\__enumext_label_style:Nnn`
`\__enumext_label_style:cvn`

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

```
455  \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
456    {
457      \tl_clear_new:N #1
458      \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
459      \tl_gset_eq:NN \g__enumext_widest_label_tl #1
460      \tl_map_inline:Nn \g__enumext_counter_styles_tl
461        {
462          \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
463          \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
464            { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
465        }
466      \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
467        { \tl_use:N \g__enumext_widest_label_tl }
468      \tl_set_eq:cN { the #2 } #1
469    }
470  \cs_generate_variant:Nn \__enumext_label_style:Nnn { cvn }
```

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

11.12 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.

```
471  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
472    {
473      \keys_define:nn { enumext / #1 }
474        {
475          font        .tl_set:c   = { l__enumext_label_font_style_#2_tl },
476          font        .value_required:n = true,
477          labelsep    .dim_set:c  = { l__enumext_labelsep_#2_dim },
478          labelsep    .initial:n  = {0.3333em},
479          labelsep    .value_required:n = true,
480          labelwidth  .dim_set:c  = { l__enumext_labelwidth_#2_dim },
481          labelwidth  .value_required:n = true,
```

```
482        wrap-label  .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
483        wrap-label  .initial:n  = {##1},
484        wrap-label  .value_required:n = true,
485        wrap-label* .code:n = {
486                              \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
487                              \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
488                          },
489        wrap-label* .value_required:n = true,
490      }
491    }
492 \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.

```
493 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
494    {
495      \keys_define:nn { enumext / #1 }
496        {
497          align .choice:,
498          align / left     .code:n =
499                              {
500                                \tl_clear:c { l__enumext_label_fill_left_#2_tl  }
501                                \tl_set:cn  { l__enumext_label_fill_right_#2_tl } { \hfill }
502                              },
503          align / right    .code:n =
504                              {
505                                \tl_set:cn  { l__enumext_label_fill_left_#2_tl  } { \hfill }
506                                \tl_clear:c { l__enumext_label_fill_right_#2_tl }
507                              },
508          align / center   .code:n =
509                              {
510                                \tl_set:cn { l__enumext_label_fill_left_#2_tl  } { \hfill }
511                                \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
512                              },
513          align / unknown .code:n =
514                              \msg_error:nneee { enumext } { unknown-choice }
515                                { align } { left, ~ right, ~  center } { \exp_not:n {##1} },
516          align .initial:n  = left,
517          align .value_required:n  = true,
518        }
519    }
520 \clist_map_inline:nn
521    {
522      {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
523    }
524    { \__enumext_tmp:nn #1 }

525 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
526    {
527      \keys_define:nn { enumext / #1 }
528        {
529          align .choice:,
530          align / left     .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
531          align / right    .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
532          align / center   .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
533          align / unknown .code:n =
534                              \msg_error:nneee { enumext } { unknown-choice }
535                                { align } { left, ~ right, ~  center } { \exp_not:n {##1} },
536          align .initial:n = left,
537          align .value_required:n = true,
538        }
539    }
540 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }
```

(*End of definition for* align.)

11.13 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.

11.13.1 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.

```
541 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
542   {
543     \keys_define:nn { enumext / #1 }
544       {
545         label .code:n   = {
546                           \__enumext_label_style:cvn { l__enumext_label_#2_tl }
547                           { l__enumext_counter_#2_tl } {##1}
548                           \dim_set_eq:cN  { l__enumext_labelwidth_#2_dim }
549                           \l__enumext_current_widest_dim
550                         },
551         label .initial:n = #3,
552         label .value_required:n = true,
553         ref    .code:n   = \__enumext_standar_ref:n {##1},
554         ref    .value_required:n = true,
555       }
556   }
557 \__enumext_tmp:nnn { level-1 } {   i } { \arabic*.}
558 \__enumext_tmp:nnn { level-2 } {  ii } { (\alph*) }
559 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
560 \__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.

```
561 \cs_new_protected:Npn \__enumext_standar_ref:n #1
562   {
563     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
564     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
565       {
566         \msg_error:nnn { enumext } { key-ref-empty } { enumext }
567       }
568       {
569         \tl_set_eq:Nc
570           \l__enumext_ref_the_count_tl { l__enumext_counter_ \__enumext_level: _tl }
571         \__enumext_regex_counter_style:
572         \tl_set_eq:Nc
573           \l__enumext_ref_the_count_tl { l__enumext_the_counter_ \__enumext_level: _tl }
574         \tl_put_right:ce { l__enumext_renew_the_count_ \__enumext_level: _tl }
575           {
576             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
577               { \exp_not:V \l__enumext_ref_key_arg_tl }
578           }
579       }
580   }
```

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

```
581 \cs_new_protected:Nn \__enumext_standar_ref:
582   {
583     \tl_if_empty:cF { l__enumext_renew_the_count_ \__enumext_level: _tl }
584       {
585         \tl_use:c { l__enumext_renew_the_count_ \__enumext_level: _tl }
586       }
587   }
```

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

11.13.2   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.

```
588 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
589   {
590     \keys_define:nn { enumext / #1 }
591       {
592         label .code:n    = {
593                             \__enumext_label_style:cvn { l__enumext_label_#2_tl }
594                               { l__enumext_counter_#2_tl } {##1}
595                             \dim_set_eq:cN  { l__enumext_labelwidth_#2_dim }
596                               \l__enumext_current_widest_dim
597                           },
598         label .initial:n = #3,
599         label .value_required:n = true,
600         ref   .code:n    = \__enumext_starred_ref:n {##1},
601         ref   .value_required:n = true,
602       }
603   }
604 \__enumext_tmp:nnn { enumext* } {  vii } { \arabic*.}
605 \__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`.

```
606 \cs_new_protected:Npn \__enumext_starred_ref:n #1
607   {
608     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
609     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
610       {
611         \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
612           {
613             \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
614           }
615           {
616             \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
617             \__enumext_regex_counter_style:
618             \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
619             \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
620               {
621                 \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
622                   { \exp_not:V \l__enumext_ref_key_arg_tl }
623               }
624           }
625       }
626     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
627       {
628         \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
629           {
630             \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
631           }
632           {
633             \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
634             \__enumext_regex_counter_style:
635             \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
636             \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
637               {
638                 \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
639                   { \exp_not:V \l__enumext_ref_key_arg_tl }
640               }
641           }
642       }
643   }
```

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.

```
644 \cs_new_protected:Nn \__enumext_starred_ref:
645   {
646     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
647       {
```

```
648          \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
649            {
650              \tl_use:N \l__enumext_renew_the_count_vii_tl
651            }
652        }
653      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
654        {
655          \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
656            {
657              \tl_use:N \l__enumext_renew_the_count_viii_tl
658            }
659        }
660    }
```

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

11.13.3 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.

```
661  \keys_define:nn { enumext / keyans }
662    {
663      label .code:n    = {
664                          \__enumext_label_style:cvn { l__enumext_label_v_tl }
665                            { l__enumext_counter_v_tl } {#1}
666                          \dim_set_eq:cN  { l__enumext_labelwidth_v_dim }
667                            \l__enumext_current_widest_dim
668                          \__enumext_label_style:cvn { l__enumext_label_vi_tl }
669                            { l__enumext_counter_vi_tl } {#1}
670                          \dim_set_eq:cN  { l__enumext_labelwidth_v_dim }
671                            \l__enumext_current_widest_dim
672                        },
673      label .initial:n = \Alph*),
674      label .value_required:n = true,
675      ref   .code:n    = \__enumext_keyans_ref:n {#1},
676      ref   .value_required:n = true,
677    }
```

(*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`.

```
678  \cs_new_protected:Npn \__enumext_keyans_ref:n #1
679    {
680      \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
681      \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
682        {
683          \msg_error:nnn { enumext } { key-ref-empty } { keyans }
684        }
685        {
686          \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
687          \__enumext_regex_counter_style:
688          \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
689          \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
690            {
691              \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
692                { \exp_not:V \l__enumext_ref_key_arg_tl }
693            }
694        }
695    }
```

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

```
696  \cs_new_protected:Nn \__enumext_keyans_ref:
697    {
698      \tl_if_empty:NF \l__enumext_renew_the_count_v_tl
699        {
700          \tl_use:N \l__enumext_renew_the_count_v_tl
701        }
702    }
```

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

11.14 Setting start and widest keys

`\__enumext_start_from:NNn`
`\__enumext_start_from:ccn`

The function `\__enumext_start_from:NNn` used by the start key 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.

```
703 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
704   {
705     \__enumext_if_is_int:nTF { #3 }
706       {
707         \int_set:Nn #2 {#3}
708       }
709       {
710         \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
711           { \int_set:Nn #2 { \int_from_alph:n {#3} } }
712         \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
713           { \int_set:Nn #2  { \int_from_roman:n {#3} } }
714       }
715   }
716 \cs_generate_variant:Nn \__enumext_start_from:NNn { ccn }
```

(*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.

```
717 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
718   {
719     \__enumext_if_is_int:nTF {#4}
720       {
721         \setcounter{enumX#1} { #4 }
722       }
723       {
724         \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
725           { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
726         \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
727           { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
728       }
729     \__enumext_label_width_by_box:cv
730       { l__enumext_labelwidth_#1_dim } { l__enumext_label_#1_tl }
731   }
732 \cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }
```

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

start
widest
`\l__enumext_start_X_int`

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

```
733 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
734   {
735     \keys_define:nn { enumext / #1 }
736       {
737         start  .code:n   = {
738                             \__enumext_start_from:ccn
739                               { l__enumext_label_#2_tl }
740                               { l__enumext_start_#2_int } {##1}
741                            },
742         start  .initial:n = 1,
743         widest .code:n    = {
744                             \__enumext_widest_from:nccn {#2}
745                               { l__enumext_label_#2_tl }
746                               { l__enumext_labelwidth_#2_dim } {##1}
747                            },
```

```
748          widest .value_required:n = true,
749          start  .value_required:n = true,
750        }
751    }
752  \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

11.15 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.

```
753  \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
754    {
755      \keys_define:nn { enumext / #1 }
756        {
757          topsep    .skip_set:c = { l__enumext_topsep_#2_skip },
758          topsep    .initial:n  = {#3},
759          topsep    .value_required:n = true,
760          partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
761          partopsep .initial:n  = {#4},
762          partopsep .value_required:n = true,
763          parsep    .skip_set:c = { l__enumext_parsep_#2_skip },
764          parsep    .initial:n  = {#5},
765          parsep    .value_required:n = true,
766          itemsep   .skip_set:c = { l__enumext_itemsep_#2_skip },
767          itemsep   .initial:n  = {#6},
768          itemsep   .value_required:n = true,
769          noitemsep .meta:n     = { itemsep = 0pt, parsep = 0pt },
770          noitemsep .value_forbidden:n = true,
771          nosep     .meta:n     = {
772                                    itemsep = 0pt, parsep= 0pt,
773                                    topsep = 0pt, partopsep = 0pt,
774                                  },
775          nosep     .value_forbidden:n = true,
776        }
777    }
```

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

```
778  \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
779    { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
780    { 4.0pt plus 2.0pt minus 1.0pt }
781  \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
782    { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
783    { 2.0pt plus 1.0pt minus 1.0pt }
784  \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
785    { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
786  \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
787    { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
788  \__enumext_tmp:nnnnnn { keyans  } { v }{ 4.0pt plus 2.0pt minus 1.0pt }
789    { 2.0pt plus 1.0pt minus 1.0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
790    { 2.0pt plus 1.0pt minus 1.0pt }
791  \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
792    { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
793    { 4.0pt plus 2.0pt minus 1.0pt }
794  \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
795    { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
796    { 2.0pt plus 1.0pt minus 1.0pt }
```

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

11.16 Setting base-fix key

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

base-fix
__enumext_nested_base_line_fix:

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

```
797  \cs_set_protected:Npn \__enumext_tmp:n #1
798    {
799      \keys_define:nn { enumext / #1 }
```

```
800        {
801            base-fix .bool_set:N = \l__enumext_base_line_fix_bool,
802            base-fix .initial:n  = false,
803            base-fix .value_forbidden:n = true,
804        }
805    }
806 \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 (§11.37) and to the function `\__enumext_parse_keys_vii:n` in the enumext* environment definition (§11.42)

```
807 \cs_new_protected:Nn \__enumext_nested_base_line_fix:
808    {
809      \bool_lazy_and:nnT
810        { \bool_if_p:N \l__enumext_standar_first_bool }
811        { \bool_if_p:N \l__enumext_base_line_fix_bool }
812        {
813          \mode_leave_vertical:
814          \vspace { -\baselineskip }
815          \keys_set:nn { enumext / level-1 }
816            {
817              topsep = 0pt, above = 0pt, above* = 0pt,
818            }
819        }
820      \bool_lazy_and:nnT
821        { \bool_if_p:N \l__enumext_starred_first_bool }
822        { \bool_if_p:N \l__enumext_base_line_fix_bool }
823        {
824          \mode_leave_vertical:
825          \vspace { -\baselineskip }
826          \keys_set:nn { enumext / enumext* }
827            {
828              topsep = 0pt, above = 0pt, above* = 0pt,
829            }
830        }
831      \bool_set_false:N \l__enumext_base_line_fix_bool
832    }
```

💣 This key is enabled by default in the command `\printkeyans` (§11.45).

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

11.17   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

```
833 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
834    {
835      \keys_define:nn { enumext / #1 }
836        {
837          itemindent    .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
838          itemindent    .value_required:n = true,
839          rightmargin   .dim_set:c = { l__enumext_rightmargin_#2_dim },
840          rightmargin   .value_required:n = true,
841          listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
842          listparindent .value_required:n = true,
843          list-offset   .dim_set:c = { l__enumext_listoffset_#2_dim },
844          list-offset   .value_required:n = true,
845          list-indent   .code:n     =
846                          \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
847                          \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
848          list-indent   .value_required:n = true,
849        }
850    }
851 \clist_map_inline:nn
852    {
853      {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
854    }
855    { \__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.

```
856  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
857    {
858      \keys_define:nn { enumext / #1 }
859        {
860          itemindent    .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
861          itemindent    .value_required:n = true,
862          rightmargin   .dim_set:c = { l__enumext_rightmargin_#2_dim },
863          rightmargin   .value_required:n = true,
864          listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
865          listparindent .value_required:n = true,
866          list-offset   .dim_set:c = { l__enumext_listoffset_#2_dim },
867          list-offset   .value_required:n = true,
868          list-indent   .meta:n    = { list-offset = ##1 },
869          list-indent   .value_required:n = true,
870        }
871    }
872  \clist_map_inline:nn
873    {
874      {enumext*}{vii}, {keyans*}{viii}
875    }
876    { \__enumext_tmp:nn #1 }
```

11.17.1 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.

```
877  \cs_set_protected:Nn \__enumext_fake_item:
878    {
879      \dim_compare:nNnT
880        { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
881        >
882        { \c_zero_dim }
883        {
884          \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
885            {
886              \exp_not:N \mode_leave_vertical:
887              \exp_not:n { \skip_horizontal:n }
888                { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
889              \ignorespaces
890            }
891        }
892    }
893  \cs_set_protected:Nn \__enumext_keyans_fake_item:
894    {
895      \dim_compare:nNnT
896        { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
897        {
898          \tl_set:Ne \l__enumext_fake_item_indent_v_tl
899            {
900              \exp_not:N \mode_leave_vertical:
901              \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
902            }
903        }
904    }
905  \cs_set_protected:Nn \__enumext_fake_item_vii:
906    {
907      \dim_compare:nNnT
908        { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
909        {
910          \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
911            {
912              \exp_not:N \mode_leave_vertical:
913              \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
914            }
915        }
916    }
917  \cs_set_protected:Nn \__enumext_fake_item_viii:
```

```
918     {
919       \dim_compare:nNnT
920         { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
921         {
922           \tl_set:Ne \l__enumext_fake_item_indent_viii_tl
923             {
924               \exp_not:N \mode_leave_vertical:
925               \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
926             }
927         }
928     }
```

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

11.18 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.

```
929 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
930   {
931     \keys_define:nn { enumext / #1 }
932       {
933         show-length .bool_set:c = { l__enumext_show_length_#2_bool },
934         show-length .initial:n  = false,
935       }
936   }
937 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

11.19 Setting before, after and first keys

before
before*
after
first

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

```
938 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
939   {
940     \keys_define:nn { enumext / # 1 }
941       {
942         before  .tl_set:c   = { l__enumext_before_no_starred_key_#2_tl },
943         before  .value_required:n = true,
944         before* .tl_set:c   = { l__enumext_before_starred_key_#2_tl },
945         before* .value_required:n = true,
946         after   .tl_set:c   = { l__enumext_after_stop_list_#2_tl },
947         after   .value_required:n = true,
948         first   .tl_set:c   = { l__enumext_after_list_args_#2_tl },
949         first   .value_required:n = true,
950       }
951   }
952 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

11.19.1 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 *second argument* of the list.

```
953 \cs_new_protected:Nn \__enumext_before_args_exec:
954   {
955     \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
956   }
```

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*⟩.

```
957 \cs_new_protected:Nn \__enumext_before_keys_exec:
958   {
959     \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
960   }
```

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

```
961  \cs_new_protected:Nn \__enumext_after_stop_list:
962    {
963      \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
964    }
```

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.

```
965  \cs_new_protected:Nn \__enumext_after_args_exec:
966    {
967      \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
968    }
```

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

11.19.2 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:

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

```
969  \cs_new_protected:Nn \__enumext_before_args_exec_v:
970    {
971      \tl_use:N \l__enumext_before_starred_key_v_tl
972    }
```

The function __enumext_before_keys_exec_v: executes the {⟨*code*⟩} set by the before key *"before"* the keyans environment is started in {⟨*arg two*⟩} of the list. The {⟨*code*⟩} is executed *"knowing"* all definition and values provides by ⟨*keys*⟩.

```
973  \cs_new_protected:Nn \__enumext_before_keys_exec_v:
974    {
975      \tl_use:N \l__enumext_before_no_starred_key_v_tl
976    }
```

The function __enumext_after_stop_list_v: executes the {⟨*code*⟩} set by the after key *"after"* the keyans environment has finished.

```
977  \cs_new_protected:Nn \__enumext_after_stop_list_v:
978    {
979      \tl_use:N \l__enumext_after_stop_list_v_tl
980    }
```

The function __enumext_after_args_exec_v: executes the {⟨*code*⟩} set by the first key after the end of {⟨*arg two*⟩} of the list defining the keyans environment, just before the first occurrence of \item.

```
981  \cs_new_protected:Nn \__enumext_after_args_exec_v:
982    {
983      \tl_use:N \l__enumext_after_list_args_v_tl
984    }
```

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

11.19.3 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:

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

```
985  \cs_new_protected:Nn \__enumext_before_args_exec_vii:
986    {
987      \tl_use:N \l__enumext_before_starred_key_vii_tl
988    }
989  \cs_new_protected:Nn \__enumext_before_args_exec_viii:
990    {
991      \tl_use:N \l__enumext_before_starred_key_viii_tl
992    }
```

The functions __enumext_before_keys_exec_vii: and __enumext_before_keys_exec_viii: executes the {⟨*code*⟩} set by the before key *"before"* in enumext* and keyans* environments is started in {⟨*arg two*⟩} of the list. The {⟨*code*⟩} is executed *"knowing"* all definition and values provides by ⟨*keys*⟩.

```
993  \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
994    {
995      \tl_use:N \l__enumext_before_no_starred_key_vii_tl
996    }
997  \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
998    {
```

```
999        \tl_use:N \l__enumext_before_no_starred_key_viii_tl
1000     }
```

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

```
1001  \cs_new_protected:Nn \__enumext_after_stop_list_vii:
1002    {
1003        \tl_use:N \l__enumext_after_stop_list_vii_tl
1004    }
1005  \cs_new_protected:Nn \__enumext_after_stop_list_viii:
1006    {
1007        \tl_use:N \l__enumext_after_stop_list_viii_tl
1008    }
```

The function `\__enumext_after_args_exec_v:` executes the `{⟨code⟩}` set by the first key after the end of `{⟨arg two⟩}` of the list defining the keyans environment, just before the first occurrence of `\item`.

```
1009  \cs_new_protected:Nn \__enumext_after_args_exec_vii:
1010    {
1011        \tl_use:N \l__enumext_after_list_args_vii_tl
1012    }
1013  \cs_new_protected:Nn \__enumext_after_args_exec_viii:
1014    {
1015        \tl_use:N \l__enumext_after_list_args_viii_tl
1016    }
```

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

11.20 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.

```
1017  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1018    {
1019        \keys_define:nn { enumext / #1 }
1020          {
1021            mini-env    .dim_set:c  = { l__enumext_minipage_right_#2_dim },
1022            mini-env    .value_required:n = true,
1023            mini-sep    .dim_set:c  = { l__enumext_minipage_hsep_#2_dim },
1024            mini-sep    .initial:n  = 0.3333em,
1025            mini-sep    .value_required:n = true,
1026            columns-sep .dim_set:c  = { l__enumext_columns_sep_#2_dim },
1027            columns-sep .value_required:n = true,
1028            columns     .int_set:c  = { l__enumext_columns_#2_int },
1029            columns     .initial:n  = 1,
1030            columns     .value_required:n = true,
1031          }
1032    }
1033  \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.

```
1034  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1035    {
1036        \keys_define:nn { enumext / #1 }
1037          {
1038            mini-right  .tl_gset:c = { g__enumext_miniright_code_#2_tl },
1039            mini-right  .value_required:n = true,
1040            mini-right* .code:n    = {
1041                                      \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
1042                                      \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1043                                    },
1044            mini-right* .value_required:n = true,
1045          }
1046    }
1047  \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }
```

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

11.21 Adjustment of vertical spaces for multicols

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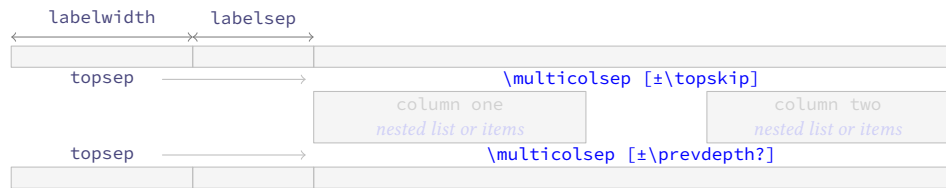
Figure 7: Representation of the vertical space in multicols for a nested level.

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

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

11.21.1 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*.

```
1048 \cs_new_protected:Nn \__enumext_multi_set_vskip:
1049   {
1050     \skip_set:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1051       {
1052         \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1053       }
1054     \skip_set:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1055       {
1056         \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1057       }
1058     \__enumext_add_pre_parsep:
1059   }
```

(*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*.

```
1060 \cs_new_protected:Nn \__enumext_add_pre_parsep:
1061   {
1062     \int_case:nn { \l__enumext_level_int }
1063       {
1064         { 2 }{
1065             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1066               {
1067                 \skip_add:Nn \l__enumext_multicols_above_ii_skip { \l__enumext_parsep_i_skip }
1068               }
1069           }
1070         { 3 }{
1071             \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1072               {
1073                 \skip_add:Nn \l__enumext_multicols_above_iii_skip { \l__enumext_parsep_ii_skip
1074               }
1075           }
1076         { 4 }{
1077             \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1078               {
1079                 \skip_add:Nn \l__enumext_multicols_above_iv_skip { \l__enumext_parsep_iii_skip
1080               }
1081           }
```

```
1082                }
1083            }
```

(*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*⟩.

```
1084 \cs_new_protected:Nn \__enumext_multi_addvspace:
1085    {
1086        \__enumext_multi_set_vskip:
1087        \mode_if_vertical:T
1088          {
1089            \skip_add:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1090              {
1091                \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1092              }
1093            \skip_add:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1094              {
1095                \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1096              }
1097          }
1098        \par\nopagebreak
1099        \addvspace{ \skip_use:c { l__enumext_multicols_above_ \__enumext_level: _skip } }
1100    }
```

(*End of definition for* __enumext_multi_addvspace:.)

11.21.2 Adjustment of vertical spaces for multicols in keyans

__enumext_keyans_multi_set_vskip: The function __enumext_keyans_multi_set_vskip: will take care of determining the *"adjusted*
__enumext_keyans_multi_addvspace: *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.

```
1101 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1102    {
1103        \skip_set:Nn \l__enumext_multicols_above_v_skip
1104          {
1105            \l__enumext_topsep_v_skip
1106          }
1107        \skip_set:Nn \l__enumext_multicols_below_v_skip
1108          {
1109            \l__enumext_topsep_v_skip
1110          }
1111    }
1112 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1113    {
1114        \__enumext_keyans_multi_set_vskip:
1115        \mode_if_vertical:T
1116          {
1117            \skip_add:Nn \l__enumext_multicols_above_v_skip
1118              {
1119                \skip_use:N \l__enumext_partopsep_v_skip
1120              }
1121            \skip_add:Nn \l__enumext_multicols_below_v_skip
1122              {
1123                \skip_use:N \l__enumext_partopsep_v_skip
1124              }
1125          }
1126        \par\nopagebreak
1127        \addvspace{ \l__enumext_multicols_above_v_skip }
1128    }
```

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

11.22 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.
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.
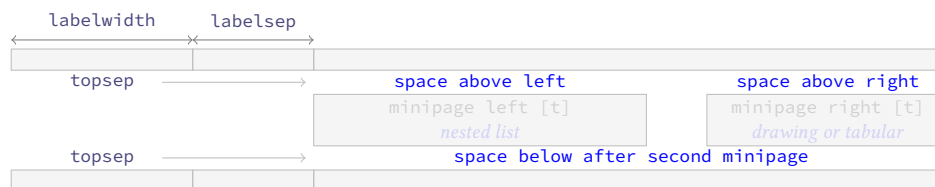
Figure 8: Representation of the `minipage` spacing adjustment for a nested level.

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.

11.22.1 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`.

```
1129  \cs_new_protected:Nn \__enumext_mini_set_vskip:
1130    {
1131      \int_compare:nNnTF
1132        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1133        {
```

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.

```
1134          \skip_if_eq:nnTF
1135            { \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1136            {
1137              \skip_set:Nn \l__enumext_minipage_left_skip
1138                {
1139                  -0.150\box_dp:N \strutbox
1140                }
1141              \skip_set:Nn \l__enumext_minipage_right_skip
1142                {
1143                  0.695\box_dp:N \strutbox
1144                }
1145              \skip_set:Nn \l__enumext_minipage_after_skip
1146                {
1147                  \box_dp:N \strutbox
1148                }
1149              \__enumext_zero_parsep:
1150            }
1151            {
1152              \skip_set:Nn \l__enumext_minipage_left_skip
1153                {
1154                  \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1155                }
1156              \skip_set:Nn \l__enumext_minipage_right_skip
1157                {
1158                  0.695\box_dp:N \strutbox
1159                }
1160              \skip_set:Nn \l__enumext_minipage_after_skip
1161                {
1162                  1.85\box_dp:N \strutbox
1163                  + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1164                }
```

```
1165                    }
1166                }
1167            {
```

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.

```
1168            \skip_if_eq:nnTF
1169              { \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip } } { \c_zero_skip }
1170              {
1171                \skip_set:Nn \l__enumext_minipage_left_skip
1172                  {
1173                    0.5\box_dp:N \strutbox
1174                    - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1175                  }
1176                \skip_set:Nn \l__enumext_minipage_right_skip
1177                  {
1178                    \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1179                  }
1180                \skip_set:Nn \l__enumext_minipage_after_skip
1181                  {
1182                    1.6\box_dp:N \strutbox
1183                  }
1184              }
1185              {
1186                \skip_set:Nn \l__enumext_minipage_left_skip
1187                  {
1188                    0.5875\box_dp:N \strutbox
1189                    - \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1190                  }
1191                \skip_set:Nn \l__enumext_minipage_right_skip
1192                  {
1193                    + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1194                    + \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1195                  }
1196                \skip_set:Nn \l__enumext_minipage_after_skip
1197                  {
1198                    0.325\box_dp:N \strutbox
1199                    + \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1200                  }
1201              }
1202            }
1203        }
```

(*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.

```
1204  \cs_new_protected:Nn \__enumext_zero_parsep:
1205    {
1206      \int_case:nn { \l__enumext_level_int }
1207        {
1208          { 2 }{
1209                  \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1210                    {
1211                      \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1212                    }
1213                }
1214          { 3 }{
1215                  \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1216                    {
1217                      \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1218                    }
1219                }
1220          { 4 }{
1221                  \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1222                    {
1223                      \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1224                    }
1225                }
```

```
1226        }
1227      }
```

(*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*.

```
1228  \cs_new_protected:Nn \__enumext_mini_addvspace:
1229    {
1230      \__enumext_mini_set_vskip:
1231      \mode_if_vertical:T
1232        {
1233          \skip_add:Nn \l__enumext_minipage_left_skip
1234            {
1235              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1236            }
1237          \skip_add:Nn \l__enumext_minipage_after_skip
1238            {
1239              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1240            }
1241        }
1242      \par\nopagebreak
1243      \addvspace { \l__enumext_minipage_left_skip }
1244    }
```

(*End of definition for* __enumext_mini_addvspace:.)

11.22.2 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 imple-
mentation of this function is the same as the one used in enumext.

```
1245  \cs_new_protected:Nn \__enumext_keyans_mini_set_vskip:
1246    {
1247      \skip_zero_new:N \l__enumext_minipage_after_skip
1248      \skip_zero_new:N \l__enumext_minipage_left_skip
1249      \skip_zero_new:N \l__enumext_minipage_right_skip
1250      \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1251        {
1252          \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1253            {
1254              \skip_set:Nn \l__enumext_minipage_left_skip { -0.25\box_dp:N \strutbox }
1255              \skip_set:Nn \l__enumext_minipage_right_skip { 0.705\box_dp:N \strutbox }
1256              \skip_set:Nn \l__enumext_minipage_after_skip { \box_dp:N \strutbox }
1257              \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1258                {
1259                  \skip_add:Nn \l__enumext_minipage_after_skip { 2.15\box_dp:N \strutbox }
1260                }
1261            }
1262            {
1263              \skip_set:Nn \l__enumext_minipage_left_skip
1264                {
1265                  \skip_use:N \l__enumext_topsep_v_skip
1266                }
1267              \skip_set:Nn \l__enumext_minipage_right_skip
1268                {
1269                  0.705\box_dp:N \strutbox
1270                }
1271              \skip_set:Nn \l__enumext_minipage_after_skip
1272                {
1273                  1.85\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1274                }
1275            }
1276        }
1277        {
1278          \skip_if_eq:nnTF { \l__enumext_topsep_v_skip } { \c_zero_skip }
1279            {
1280              \skip_set:Nn \l__enumext_minipage_left_skip
1281                {
```

```
1282              0.5\box_dp:N \strutbox
1283              + \l__enumext_partopsep_v_skip
1284            }
1285        \skip_set:Nn \l__enumext_minipage_right_skip
1286          {
1287            \l__enumext_partopsep_v_skip
1288          }
1289        \skip_set:Nn \l__enumext_minipage_after_skip { 1.6\box_dp:N \strutbox }
1290      }
1291      {
1292        \skip_set:Nn \l__enumext_minipage_left_skip
1293          {
1294            0.5875\box_dp:N \strutbox - \l__enumext_partopsep_v_skip
1295          }
1296        \skip_set:Nn \l__enumext_minipage_right_skip
1297          {
1298            \l__enumext_topsep_v_skip + \l__enumext_partopsep_v_skip
1299          }
1300        \skip_set:Nn \l__enumext_minipage_after_skip
1301          {
1302            0.325\box_dp:N \strutbox + \l__enumext_topsep_v_skip
1303          }
1304      }
1305    }
1306  }
```

(*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.

```
1307 \cs_new_protected:Nn \__enumext_keyans_mini_addvspace:
1308   {
1309     \__enumext_keyans_mini_set_vskip:
1310     \mode_if_vertical:T
1311       {
1312         \skip_add:Nn \l__enumext_minipage_left_skip
1313           {
1314             \l__enumext_partopsep_v_skip
1315           }
1316         \skip_add:Nn \l__enumext_minipage_after_skip
1317           {
1318             \l__enumext_partopsep_v_skip
1319           }
1320       }
1321     \par\nopagebreak
1322     \addvspace { \l__enumext_minipage_left_skip }
1323   }
```

(*End of definition for* __enumext_keyans_mini_addvspace:.)

11.22.3   Adjustment of vertical spaces for minipage in enumext* and keyans*

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

```
1324 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1325   {
1326     \skip_zero_new:N \l__enumext_minipage_left_skip
1327     \skip_gzero_new:N \g__enumext_minipage_right_skip
1328     \skip_gzero_new:N \g__enumext_minipage_after_skip
1329     \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1330       {
1331         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1332         \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1333       }
1334       {
1335         \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1336         \skip_gset:Nn \g__enumext_minipage_right_skip
```

```
1337              {
1338                \l__enumext_topsep_vii_skip
1339              }
1340          \skip_gset:Nn \g__enumext_minipage_after_skip
1341            {
1342              0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1343            }
1344        }
1345    }
1346 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1347    {
1348      \skip_zero_new:N \l__enumext_minipage_after_skip
1349      \skip_zero_new:N \l__enumext_minipage_left_skip
1350      \skip_zero_new:N \l__enumext_minipage_right_skip
1351      \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1352        {
1353          \skip_set:Nn \l__enumext_minipage_left_skip
1354            {
1355              0.5\box_dp:N \strutbox
1356            }
1357          \skip_set:Nn \l__enumext_minipage_right_skip
1358            {
1359              \l__enumext_partopsep_viii_skip
1360            }
1361          \skip_set:Nn \l__enumext_minipage_after_skip
1362            {
1363              1.6\box_dp:N \strutbox
1364            }
1365        }
1366        {
1367          \skip_set:Nn \l__enumext_minipage_left_skip
1368            {
1369              0.5875\box_dp:N \strutbox
1370            }
1371          \skip_set:Nn \l__enumext_minipage_right_skip
1372            {
1373              \l__enumext_topsep_viii_skip
1374            }
1375          \skip_set:Nn \l__enumext_minipage_after_skip
1376            {
1377              0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1378            }
1379        }
1380    }
```

(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.

```
1381 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1382    {
1383      \__enumext_mini_set_vskip_vii:
1384      \par\nopagebreak
1385      \addvspace { \l__enumext_minipage_left_skip }
1386    }
1387 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1388    {
1389      \__enumext_mini_set_vskip_viii:
1390      \par\nopagebreak
1391      \addvspace { \l__enumext_minipage_left_skip }
1392    }
```

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

11.22.4   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 from being executed inside the keyanspic environment, then we call the internal functions for the enumext and keyans environments.

```
1393 \NewDocumentCommand \miniright { s }
1394   {
1395     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1396       {
1397         \msg_error:nnn { enumext } { wrong-miniright-place }
1398       }
1399     \int_compare:nNnT { \l__enumext_level_int } = { 0 }
1400       {
1401         \msg_error:nnn { enumext } { wrong-miniright-place }
1402       }
1403     \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
1404       {
1405         \__enumext_keyans_mini_right_cmd:n {#1}
1406       }
1407       { \__enumext_mini_right_cmd:n {#1} }
1408   }
```

(*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.

```
1409 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1410   {
1411     \dim_compare:nNnTF
1412       { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1413       {
1414         \__enumext_multicols_stop:
1415         \end{__enumext_mini_env*}
1416         \hfill
1417         \begin{__enumext_mini_env*}
1418           { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } }
1419           \par\addvspace { \l__enumext_minipage_right_skip }
1420           \bool_if:nF {#1}
1421             {
1422               \centering
1423             }
1424           \int_gzero:N \g__enumext_minipage_stat_int
1425       }
1426       { \msg_error:nnn { enumext } { wrong-miniright-use } }
1427   }
```

(*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.

```
1428 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1429   {
1430     \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1431       {
1432         \__enumext_keyans_multicols_stop:
1433         \end{__enumext_mini_env*}
1434         \hfill
1435         \begin{__enumext_mini_env*}{ \l__enumext_minipage_right_v_dim }
1436           \par\addvspace { \l__enumext_minipage_right_skip }
1437           \bool_if:nF {#1}
1438             {
1439               \centering
1440             }
```

```
1441        \int_gzero:N \g__enumext_minipage_stat_int
1442      }
1443    { \msg_error:nnn { enumext } { wrong-miniright-use } }
1444  }
```

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

11.23   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.

```
1445 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1446   {
1447     \keys_define:nn { enumext / #1 }
1448       {
1449         above  .skip_set:c = { l__enumext_vspace_above_#2_skip },
1450         above  .value_required:n = true,
1451         above* .code:n     = \bool_set_true:c { l__enumext_vspace_a_star_#2_bool }
1452                              \keys_set:nn { enumext / #1 } { above = {##1} },
1453         above* .value_required:n = true,
1454         below  .skip_set:c = { l__enumext_vspace_below_#2_skip },
1455         below  .value_required:n = true,
1456         below* .code:n     = \bool_set_true:c { l__enumext_vspace_b_star_#2_bool }
1457                              \keys_set:nn { enumext / #1 } { below = {##1} },
1458         below* .value_required:n = true,
1459       }
1460   }
1461 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

11.23.1   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.

```
1462 \cs_new_protected:Nn \__enumext_vspace_above:
1463   {
1464     \skip_if_eq:nnF
1465       { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1466       {
1467         \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
1468           {
1469             \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1470           }
1471           {
1472             \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1473           }
1474       }
1475   }
```

(*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.

```
1476 \cs_new_protected:Nn \__enumext_vspace_below:
1477   {
1478     \skip_if_eq:nnF
1479       { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1480       {
1481         \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1482           {
1483             \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1484           }
1485           {
1486             \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1487           }
1488       }
1489   }
```

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

11.23.2 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.

```
1490 \cs_new_protected:Nn \__enumext_vspace_above_v:
1491   {
1492     \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1493       {
1494         \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1495           {
1496             \vspace*{ \l__enumext_vspace_above_v_skip }
1497           }
1498           { \vspace { \l__enumext_vspace_above_v_skip } }
1499       }
1500   }
```

(*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.

```
1501 \cs_new_protected:Nn \__enumext_vspace_below_v:
1502   {
1503     \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1504       {
1505         \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1506           {
1507             \vspace*{ \l__enumext_vspace_below_v_skip }
1508           }
1509           { \vspace { \l__enumext_vspace_below_v_skip } }
1510       }
1511   }
```

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

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

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

```
1512 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1513   {
1514     \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1515       {
1516         \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1517           {
1518             \vspace*{ \l__enumext_vspace_above_vii_skip }
1519           }
1520           { \vspace { \l__enumext_vspace_above_vii_skip } }
1521       }
1522   }
1523 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1524   {
1525     \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1526       {
1527         \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1528           {
1529             \vspace*{ \l__enumext_vspace_above_viii_skip }
1530           }
1531           { \vspace { \l__enumext_vspace_above_viii_skip } }
1532       }
1533   }
```

(*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.

```
1534 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1535   {
1536     \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1537       {
1538         \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1539           {
```

```
1540              \vspace*{ \l__enumext_vspace_below_vii_skip }
1541            }
1542            { \vspace { \l__enumext_vspace_below_vii_skip } }
1543        }
1544    }
1545 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1546    {
1547      \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1548        {
1549          \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1550            {
1551              \vspace*{ \l__enumext_vspace_below_viii_skip }
1552            }
1553            { \vspace { \l__enumext_vspace_below_viii_skip } }
1554        }
1555    }
```

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

11.24 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*`.

```
1556 \cs_set_protected:Npn \__enumext_tmp:n #1
1557    {
1558      \keys_define:nn { enumext / #1 }
1559        {
1560          series  .str_set:N = \l__enumext_series_str,
1561          series  .value_required:n = true,
1562          resume  .code:n = \__enumext_resume_series:n {##1},
1563          resume* .code:n = \__enumext_resume_starred:,
1564          resume* .value_forbidden:n = true,
1565        }
1566    }
1567 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

11.24.1 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.

```
1568 \cs_new:Npn \__enumext_filter_series:n #1
1569    {
1570      \use:e
1571        {
1572          \keyval_parse:NNn
1573            \__enumext_filter_series_key:n
1574            \__enumext_filter_series_pair:nn {#1}
1575        }
1576    }
```

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.

```
1577 \cs_new:Npn \__enumext_filter_series_key:n #1
1578    {
1579      \str_case:nnF {#1}
1580        {
1581          { resume   } {}
1582          { resume*  } {}
1583          { base-fix } {}
1584        }
1585        { , { \exp_not:n {#1} } }
1586    }
```

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`, `save-ans` and `save-key` keys.

```
1587 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1588    {
```

```
1589        \str_case:nnF {#1}
1590          {
1591            { series } {}
1592            { resume } {}
1593            { start  } {}
1594            { save-ans } {}
1595            { save-key } {}
1596          }
1597          { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1598      }
```

(*End of definition for* __enumext_filter_series:n, __enumext_filter_series_key:n, *and* __enumext_filter_se-
ries_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 (§11.37) and to the function __enumext_parse_keys_vii:n in
the enumext* environment definition (§11.42).

```
1599   \cs_new_protected:Npn \__enumext_parse_series:n #1
1600     {
1601       \str_if_empty:NTF \l__enumext_series_str
1602         {
1603           \bool_if:NF \l__enumext_resume_active_bool
1604             {
1605               \__enumext_resume_last:n {#1}
1606             }
1607         }
1608         {
1609           \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str _tl }
1610           \tl_gset:ce { g__enumext_series_ \l__enumext_series_str _tl }
1611             { \__enumext_filter_series:n {#1} }
1612           \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str _int }
1613             {
1614               \int_new:c { g__enumext_series_ \l__enumext_series_str _int }
1615             }
1616         }
1617     }
```

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* envi-
ronment. 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.

```
1618   \cs_new_protected:Npn \__enumext_resume_last:n #1
1619     {
1620       \bool_if:NT \l__enumext_standar_first_bool
1621         {
1622           \tl_gclear:N \g__enumext_standar_series_tl
1623           \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1624         }
1625       \bool_if:NT \l__enumext_starred_first_bool
1626         {
1627           \tl_gclear:N \g__enumext_starred_series_tl
1628           \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1629         }
1630     }
```

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

11.24.2 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 (§11.37) and the enumext* environment definition (§11.42).

```
1631  \cs_new_protected:Nn \__enumext_resume_save_counter:
1632    {
1633      \bool_if:NT \g__enumext_standar_bool
1634        {
1635          \tl_if_empty:NF \l__enumext_series_str
1636            {
1637              \int_gset_eq:cN
1638                { g__enumext_series_ \l__enumext_series_str _int } \value{enumXi}
1639            }
1640          \tl_if_empty:NTF \l__enumext_resume_name_tl
1641            {
1642              \str_if_empty:NT \l__enumext_series_str
1643                {
1644                  \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1645                }
1646            }
1647            {
1648              \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1649                {
1650                  \int_gset_eq:cN
1651                    { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXi}
1652                }
1653            }
1654          \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1655            {
1656              \int_gset_eq:cN
1657                { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1658            }
1659        }
1660      \bool_if:NT \g__enumext_starred_bool
1661        {
1662          \tl_if_empty:NF \l__enumext_series_str
1663            {
1664              \int_gset_eq:cN
1665                { g__enumext_series_ \l__enumext_series_str _int } \value{enumXvii}
1666            }
1667          \tl_if_empty:NTF \l__enumext_resume_name_tl
1668            {
1669              \str_if_empty:NT \l__enumext_series_str
1670                {
1671                  \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1672                }
1673            }
1674            {
1675              \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1676                {
1677                  \int_gset_eq:cN
1678                    { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXvii}
1679                }
1680            }
1681          \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1682            {
1683              \int_gset_eq:cN
1684                { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1685            }
1686        }
1687    }
```

(*End of definition for* __enumext_resume_save_counter:.)

11.24.3 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.

```
1688 \cs_new_protected:Npn \__enumext_resume_series:n #1
1689   {
1690     \tl_if_empty:nTF {#1}
1691       {
1692         \__enumext_resume_counter:n { }
1693       }
1694       {
1695         \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1696           {
1697             \__enumext_resume_counter:n {#1}
1698             \bool_if:NT \g__enumext_standar_bool
1699               {
1700                 \keys_set:nv { enumext / level-1 }
1701                   { g__enumext_series_ \tl_to_str:n {#1} _tl }
1702               }
1703             \bool_if:NT \g__enumext_starred_bool
1704               {
1705                 \keys_set:nv { enumext / enumext* }
1706                   { g__enumext_series_ \tl_to_str:n {#1} _tl }
1707               }
1708           }
1709           {
1710             \bool_if:NT \g__enumext_standar_bool
1711               {
1712                 \msg_error:nnn { enumext } { unknown-series } {#1}
1713               }
1714             \bool_if:NT \g__enumext_starred_bool
1715               {
1716                 \msg_error:nnn { enumext } { unknown-series } {#1}
1717               }
1718           }
1719       }
1720   }
```

(*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.

```
1721 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1722   {
1723     \bool_set_true:N \l__enumext_resume_active_bool
1724     \tl_set:Nn \l__enumext_resume_name_tl {#1}
1725     \tl_if_empty:NTF \l__enumext_resume_name_tl
1726       {
1727         \__enumext_resume_counter:
1728       }
1729       {
1730         \__enumext_resume_counter_series:
1731       }
1732     \__enumext_resume_counter_save_ans:
1733   }
```

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.

```
1734 \cs_new_protected:Nn \__enumext_resume_counter:
1735   {
1736     \bool_if:NT \g__enumext_standar_bool
1737       {
1738         \int_gincr:N \g__enumext_resume_int
1739         \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1740       }
1741     \bool_if:NT \g__enumext_starred_bool
1742       {
1743         \int_gincr:N \g__enumext_resume_vii_int
```

```
1744        \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1745      }
1746    }
```

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.

```
1747  \cs_new_protected:Nn \__enumext_resume_counter_series:
1748    {
1749      \bool_if:NT \g__enumext_standar_bool
1750        {
1751          \int_set:Nn \l__enumext_start_i_int
1752            {
1753              \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1754            }
1755        }
1756      \bool_if:NT \g__enumext_starred_bool
1757        {
1758          \int_set:Nn \l__enumext_start_vii_int
1759            {
1760              \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1761            }
1762        }
1763    }
```

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.

```
1764  \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1765    {
1766      \bool_lazy_and:nnT
1767        { \bool_if_p:N \l__enumext_standar_first_bool }
1768        { \bool_if_p:N \l__enumext_store_active_bool }
1769        {
1770          \int_set:Nn \l__enumext_start_i_int
1771            {
1772              \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1773            }
1774        }
1775      \bool_lazy_and:nnT
1776        { \bool_if_p:N \l__enumext_starred_first_bool }
1777        { \bool_if_p:N \l__enumext_store_active_bool }
1778        {
1779          \int_set:Nn \l__enumext_start_vii_int
1780            {
1781              \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1782            }
1783        }
1784    }
```

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

11.24.4 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.

```
1785  \cs_new_protected:Nn \__enumext_resume_starred:
1786    {
1787      \bool_if:NT \g__enumext_standar_bool
1788        {
1789          \tl_if_empty:NF \g__enumext_standar_series_tl
1790            {
1791              \__enumext_resume_counter:n { }
1792              \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1793            }
1794        }
1795      \bool_if:NT \g__enumext_starred_bool
1796        {
1797          \tl_if_empty:NF \g__enumext_starred_series_tl
1798            {
```

```
1799          \__enumext_resume_counter:n { }
1800          \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1801        }
1802      }
1803    }
```

(*End of definition for* __enumext_resume_starred:.)

11.25 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.

11.25.1 Setting save-ans key

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

```
1804 \cs_set_protected:Npn \__enumext_tmp:n #1
1805   {
1806     \keys_define:nn { enumext / #1 }
1807       {
1808         save-ans .code:n = \__enumext_storing_set:n {##1},
1809         save-ans .value_required:n = true,
1810       }
1811   }
1812 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

11.25.2 Internal functions for save-ans key

__enumext_start_save_ans_msg:
__enumext_stop_save_ans_msg:

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

```
1813 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
1814   {
1815     \msg_term:nnVV { enumext } { save-ans-log }
1816       \g__enumext_envir_name_tl \l__enumext_store_name_tl
1817   }
1818 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
1819   {
1820     \msg_term:nnVV { enumext } { save-ans-log-hook }
1821       \g__enumext_envir_name_tl \g__enumext_store_name_tl
1822   }
```

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

__enumext_storing_set:n
__enumext_storing_exec:

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

```
1823 \cs_new_protected:Npn \__enumext_storing_set:n #1
1824   {
1825     \tl_set:Ne \l__enumext_store_name_tl {#1}
1826     \tl_if_empty:NTF \l__enumext_store_name_tl
1827       {
1828         \bool_lazy_or:nnT
1829           { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1830           {
1831             \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
1832           }
1833       }
1834       {
1835         \bool_lazy_or:nnT
1836           { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1837           {
1838             \__enumext_start_save_ans_msg:
1839             \__enumext_storing_exec:
1840           }
1841       }
1842   }
```

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* (§11.30). 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*.

```
1843  \cs_new_protected:Nn \__enumext_storing_exec:
1844    {
1845      \bool_set_true:N \l__enumext_store_active_bool
1846      \bool_set_true:N \l__enumext_check_answers_bool
1847      \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1848      \__enumext_anskey_env_make:V \l__enumext_store_name_tl
1849      \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1850        {
1851          \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
1852          \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1853        }
1854      \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1855        {
1856          \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
1857          \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
1858        }
1859      \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
1860        {
1861          \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
1862          \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
1863        }
1864    }
```

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

11.25.3 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.

11.25.4 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.

```
1865  \cs_set_protected:Npn \__enumext_tmp:n #1
1866    {
1867      \keys_define:nn { enumext / #1 }
1868        {
1869          check-ans .bool_set:N = \l__enumext_check_ans_key_bool,
1870          check-ans .initial:n  = false,
1871          check-ans .value_required:n = true,
1872          no-store  .code:n = {
1873                                \bool_set_false:N \l__enumext_check_answers_bool
1874                                \bool_set_false:N \l__enumext_check_ans_key_bool
1875                              },
1876          no-store  .value_forbidden:n = true,
1877        }
1878    }
1879  \clist_map_inline:nn
1880    {
1881      level-1, level-2, level-3, level-4, enumext*
1882    }
1883    { \__enumext_tmp:n {#1} }
```

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

11.25.5 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.

```
1884  \cs_new_protected:Nn \__enumext_check_ans_active:
1885    {
1886      \tl_if_empty:NF \l__enumext_store_name_tl
1887        {
1888          \bool_if:NT \l__enumext_check_answers_bool
1889            {
1890              \__enumext_check_ans_level:
1891            }
1892        }
1893    }
```

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"*.

```
1894  \cs_new_protected:Nn \__enumext_check_ans_level:
1895    {
1896      \int_case:nn { \l__enumext_level_int }
1897        {
1898          { 1 }{
1899                \bool_lazy_all:nT
1900                  {
1901                    { \bool_if_p:N \g__enumext_starred_bool }
1902                    { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
1903                  }
1904                  {
1905                    \int_gdecr:N \g__enumext_item_number_int
1906                    \bool_set_false:N \l__enumext_item_number_bool
1907                  }
1908              }
1909          { 2 }{
1910                \int_gdecr:N \g__enumext_item_number_int
1911                \bool_set_false:N \l__enumext_item_number_bool
1912              }
1913          { 3 }{
1914                \int_gdecr:N \g__enumext_item_number_int
1915                \bool_set_false:N \l__enumext_item_number_bool
1916              }
1917          { 4 }{
1918                \int_gdecr:N \g__enumext_item_number_int
1919                \bool_set_false:N \l__enumext_item_number_bool
```

```
1920                    }
1921              }
```

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.

```
1922        \int_case:nn { \l__enumext_level_h_int }
1923          {
1924            { 1 }{
1925                  \bool_lazy_all:nT
1926                    {
1927                      { \bool_if_p:N \g__enumext_standar_bool }
1928                      { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
1929                    }
1930                    {
1931                      \int_gdecr:N \g__enumext_item_number_int
1932                      \bool_set_false:N \l__enumext_item_number_bool
1933                    }
1934                }
1935            }
1936          }
```

(*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.

```
1937  \cs_new_protected:Nn \__enumext_check_ans_key_hook:
1938    {
1939      \bool_lazy_and:nnT
1940        { \bool_if_p:N \l__enumext_check_ans_key_bool }
1941        { \bool_if_p:N \g__enumext_standar_bool }
1942        {
1943          \bool_gset_true:N \g__enumext_check_ans_key_bool
1944        }
1945      \bool_lazy_and:nnT
1946        { \bool_if_p:N \l__enumext_check_ans_key_bool }
1947        { \bool_if_p:N \g__enumext_starred_bool }
1948        {
1949          \bool_gset_true:N \g__enumext_check_ans_key_bool
1950        }
1951    }
```

(*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:.

```
1952  \cs_new_protected:Nn \__enumext_item_answer_diff:
1953    {
1954      \int_gset:Nn \g__enumext_item_answer_diff_int
1955        {
1956          \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
1957        }
1958    }
```

(*End of definition for* __enumext_item_answer_diff:.)

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

```
1959  \cs_new_protected:Nn \__enumext_check_ans_show:
1960    {
1961      \int_case:nn { \g__enumext_item_answer_diff_int }
1962        {
1963          { -1 }{ \__enumext_check_ans_msg_less:     }
1964          {  0 }{ \__enumext_check_ans_msg_same_ok: }
1965          {  1 }{ \__enumext_check_ans_msg_greater: }
1966        }
1967    }
```

```
1968 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
1969   {
1970     \msg_warning:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
1971       { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1972   }
1973 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
1974   {
1975     \msg_term:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
1976       { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1977   }
1978 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
1979   {
1980     \msg_warning:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
1981       { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1982   }
```

(*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:.

```
1983 \cs_new_protected:Nn \__enumext_check_ans_log:
1984   {
1985     \int_case:nn { \g__enumext_item_answer_diff_int }
1986       {
1987         { -1 }{ \__enumext_check_ans_log_msg_less:     }
1988         {  0 }{ \__enumext_check_ans_log_msg_same_ok: }
1989         {  1 }{ \__enumext_check_ans_log_msg_greater: }
1990       }
1991   }
1992 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
1993   {
1994     \msg_log:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
1995       { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
1996   }
1997 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
1998   {
1999     \msg_log:nneee { enumext } { items-same-answer }  { \g__enumext_store_name_tl }
2000       { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2001   }
2002 \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
2003   {
2004     \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2005       { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2006   }
```

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

11.25.6 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.

```
2007 \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
2008   {
2009     \int_compare:nNnT
2010       { \g__enumext_check_starred_cmd_int } = { 0 }
2011       {
2012         \msg_warning:nnnV
2013           { enumext } { missing-starred }{ #1 } \l__enumext_check_start_line_env_tl
2014       }
2015     \int_compare:nNnT
2016       { \g__enumext_check_starred_cmd_int } > { 1 }
2017       {
2018         \msg_warning:nnnV
2019           { enumext } { many-starred }{ #1 } \l__enumext_check_start_line_env_tl
2020       }
2021     \int_gzero:N \g__enumext_check_starred_cmd_int
2022     \tl_clear:N \l__enumext_check_start_line_env_tl
2023   }
```

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

11.26 Keys and functions associated with storage

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

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

```
2024 \cs_set_protected:Npn \__enumext_tmp:n #1
2025   {
2026     \keys_define:nn { enumext / #1 }
2027       {
2028         wrap-ans   .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
2029         wrap-ans   .initial:n =
2030                    {
2031                      \fbox{\parbox[t]{\dimeval{\itemwidth -2\fboxsep -2\fboxrule}}{##1}}
2032                    },
2033         wrap-ans   .value_required:n = true,
2034         wrap-opt   .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
2035         wrap-opt   .initial:n = [{##1}],
2036         wrap-opt   .value_required:n = true,
2037         save-sep   .tl_set:N  = \l__enumext_store_keyans_item_opt_sep_tl,
2038         save-sep   .initial:n = {, ~ },
2039         save-sep   .value_required:n = true,
2040         mark-ans   .tl_set:N  = \l__enumext_mark_answer_sym_tl,
2041         mark-ans   .initial:n = \textasteriskcentered,
2042         mark-ans   .value_required:n = true,
2043         mark-pos   .choice:,
2044         mark-pos / left     .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2045         mark-pos / right    .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2046         mark-pos / unknown .code:n =
2047                            \msg_error:nneee { enumext } { unknown-choice }
2048                              { mark-pos } { left, ~ right } { \exp_not:n {##1} },
2049         mark-pos   .initial:n = right,
2050         mark-pos   .value_required:n = true,
2051         show-ans   .bool_set:N = \l__enumext_show_answer_bool,
2052         show-ans   .initial:n  = false,
2053         show-ans   .value_required:n = true,
2054         show-pos   .bool_set:N = \l__enumext_show_position_bool,
2055         show-pos   .initial:n  = false,
2056         show-pos   .value_required:n = true,
2057         mark-ref   .tl_set:N   = \l__enumext_mark_ref_sym_tl,
2058         mark-ref   .initial:n  = \textasteriskcentered,
2059         mark-ref   .value_required:n = true,
2060         save-ref   .bool_set:N = \l__enumext_store_ref_key_bool,
2061         save-ref   .initial:n  = false,
2062         save-ref   .value_required:n = true,
2063       }
2064   }
2065 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

mark-pos
show-ans
show-pos

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

```
2066 \cs_set_protected:Npn \__enumext_tmp:n #1
2067   {
2068     \keys_define:nn { enumext / #1 }
2069       {
2070         mark-pos .choice:,
2071         mark-pos / left  .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2072         mark-pos / right .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2073         mark-pos .initial:n = right,
2074         mark-pos .value_required:n  = true,
2075         show-ans .bool_set:N = \l__enumext_show_answer_bool,
2076         show-ans .initial:n  = false,
2077         show-ans .value_required:n = true,
2078         show-pos .bool_set:N = \l__enumext_show_position_bool,
2079         show-pos .initial:n  = false,
2080         show-pos .value_required:n = true,
2081       }
2082   }
2083 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }
```

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

11.26.1 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*⟩.

```
2084 \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2085   {
2086     \bool_if:cF { l__enumext_store_save_key_ \__enumext_level: _bool }
2087       {
2088         \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2089         \tl_set:ce
2090           { l__enumext_store_save_key_ \__enumext_level: _tl }
2091           { \__enumext_filter_save_key:n {#1} }
2092       }
2093   }
2094 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2095   {
2096     \bool_if:NF \l__enumext_store_save_key_vii_bool
2097       {
2098         \tl_clear:N \l__enumext_store_save_key_vii_tl
2099         \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2100       }
2101   }
```

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

11.26.2 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.

```
2102 \cs_set_protected:Npn \__enumext_tmp:n #1
2103   {
2104     \keys_define:nn { enumext / enumext* }
2105       {
2106         save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2107         save-key .value_required:n = true,
2108       }
2109     \keys_define:nn { enumext / #1 }
2110       {
2111         save-key .code:n = \__enumext_parse_save_key:n {##1},
2112         save-key .value_required:n = true,
2113       }
2114   }
2115 \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*`.

```
2116 \cs_new_protected:Npn \__enumext_parse_save_key:n #1
2117   {
2118     \bool_set_true:c { l__enumext_store_save_key_ \__enumext_level: _bool }
2119     \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2120     \tl_set:ce
2121       { l__enumext_store_save_key_ \__enumext_level: _tl }
2122       { \__enumext_filter_save_key:n {#1} }
2123   }
```

```
2124  \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2125    {
2126      \bool_set_true:N \l__enumext_store_save_key_vii_bool
2127      \tl_clear:N \l__enumext_store_save_key_vii_tl
2128      \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2129    }
```

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

11.26.3 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.

```
2130  \cs_new:Npn \__enumext_filter_save_key:n #1
2131    {
2132      \use:e
2133        {
2134          \keyval_parse:NNn
2135            \__enumext_filter_save_key_key:n
2136            \__enumext_filter_save_key_pair:nn {#1}
2137        }
2138    }
```

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.

```
2139  \cs_new:Npn \__enumext_filter_save_key_key:n #1
2140    {
2141      \str_case:nnF {#1}
2142        {
2143          { resume } {} { resume* } {} { no-store } {} { base-fix } {}
2144        }
2145        { , { \exp_not:n {#1} } }
2146    }
```

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.

```
2147  \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2148    {
2149      \str_case:nnF {#1}
2150        {
2151          { series   } {} { resume    } {} { save-ans } {} { save-ref   } {}
2152          { save-key } {} { check-ans } {} { show-ans } {} { show-pos   } {}
2153          { wrap-ans } {} { mark-ans  } {} { wrap-opt } {} { save-sep   } {}
2154          { mark-ref } {} { mini-env  } {} { mini-sep } {} { mini-right } {}
2155          { mini-right* } {}
2156        }
2157        { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2158    }
```

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

11.26.4 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.

```
2159  \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2160    {
2161      \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2162        {
2163          \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2164        }
2165        { #1 }
2166    }
2167  \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V, e }
```

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

11.26.5 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.

```
2168 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2169   {
2170     \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
2171   }
2172 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V, e }
```

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

11.26.6 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.

```
2173 \cs_new_protected:Nn \__enumext_store_level_open:
2174   {
2175     \bool_if:NT \l__enumext_check_answers_bool
2176       {
2177         \tl_if_empty:cTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2178           {
2179             \__enumext_store_addto_seq:n
2180               {
2181                 \item \begin{enumext}
2182               }
2183           }
2184           {
2185             \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2186               {
2187                 \item \begin{enumext} [
2188               }
2189             \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2190               {
2191                 ]
2192               }
2193             \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2194           }
2195       }
2196   }
2197 \cs_new_protected:Nn \__enumext_store_level_close:
2198   {
2199     \bool_if:NT \l__enumext_check_answers_bool
2200       {
2201         \__enumext_store_addto_seq:n { \end{enumext} }
2202       }
2203   }
```

(*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.

```
2204 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2205   {
2206     \bool_if:NT \l__enumext_check_answers_bool
2207       {
2208         \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2209           {
2210             \__enumext_store_addto_seq:n
2211               {
2212                 \item \begin{enumext*}
2213               }
2214           }
2215           {
2216             \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2217               {
2218                 \item \begin{enumext*}[
2219               }
2220             \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
```

```
2221                {
2222                  ]
2223                }
2224              \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2225            }
2226          }
2227      }
2228  \cs_new_protected:Nn \__enumext_store_level_close_vii:
2229    {
2230      \bool_if:NT \l__enumext_check_answers_bool
2231        {
2232          \__enumext_store_addto_seq:n { \end{enumext*} }
2233        }
2234    }
```

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

11.26.7 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

```
2235  \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2236    {
2237      \mode_leave_vertical:
2238      \skip_horizontal:n { -\dim_use:N #2 }
2239      \makebox[0pt][ r ]
2240        {
2241          \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2242            {
2243              \tl_use:N \l__enumext_mark_answer_sym_tl
2244            }
2245        }
2246      \skip_horizontal:n { \dim_use:N #2 }
2247    }
2248  \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }
```

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

11.27 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.

```
2249  \cs_new_protected:Nn \__enumext_store_internal_ref:
2250    {
2251      \cs_set_protected:Npn \__enumext_tmp:n ##1
2252        {
2253          \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2254          \tl_reverse:c { l__enumext_label_copy_##1_tl }
2255          \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2256          \tl_reverse:c { l__enumext_label_copy_##1_tl }
2257        }
2258      \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2259      \cs_set:Npn \__enumext_tmp:n ##1
2260        { . \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.

```
2261      \bool_lazy_all:nT
2262        {
2263          { \bool_if_p:N \g__enumext_starred_bool }
2264          { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2265        }
2266        {
2267          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
```

```
2268              { \tl_use:N \l__enumext_label_copy_vii_tl }
2269          }
2270      \bool_lazy_all:nT
2271          {
2272              { \bool_if_p:N \l__enumext_standar_bool }
2273              { \bool_if_p:N \g__enumext_starred_bool }
2274              { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2275          }
2276          {
2277              \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2278                  {
2279                      \tl_use:N \l__enumext_label_copy_vii_tl
2280                      \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2281                  }
2282          }
```

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.

```
2283      \bool_lazy_all:nT
2284          {
2285              { \bool_if_p:N \l__enumext_standar_bool }
2286              { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2287              { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2288              { \bool_not_p:n { \l__enumext_starred_bool } }
2289          }
2290          {

2291              \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2292                  {
2293                      \tl_use:N \l__enumext_label_copy_i_tl
2294                      \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2295                  }
2296          }
2297      \cs_set:Npn \__enumext_tmp:n ##1
2298          { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } }
2299      \bool_lazy_all:nT
2300          {
2301              { \bool_if_p:N \l__enumext_standar_bool }
2302              { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2303              { \bool_not_p:n { \g__enumext_starred_bool } }
2304              { \int_compare_p:nNn { \l__enumext_level_h_int } > { 0 } }
2305          }
2306          {
2307              \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2308                  {
2309                      \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2310                      . \tl_use:N \l__enumext_label_copy_vii_tl
2311                  }
2312          }
```

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

```
2313      \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2314          {
2315              \l__enumext_store_name_tl \c_colon_str
2316              \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2317          }
```

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.

```
2318      \tl_put_right:Ne \l__enumext_write_aux_file_tl
2319          {
2320              \__enumext_newlabel:nn
2321                  { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2322                  { \l__enumext_newlabel_arg_two_tl }
2323          }
2324      \l__enumext_write_aux_file_tl
2325  }
```

(*End of definition for* __enumext_store_internal_ref:.)

11.28 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⟩.

```
2326  \cs_new_protected:Npn \__enumext_store_anskey_code:n #1
2327    {
2328      \int_gincr:N \g__enumext_item_anskey_int
2329      \__enumext_store_addto_prop:n {#1}
2330      \bool_if:NT \l__enumext_store_ref_key_bool
2331        {
2332          \__enumext_store_internal_ref:
2333        }
2334      \__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.

```
2335      \tl_clear:N \l__enumext_store_anskey_arg_tl
2336      \bool_lazy_and:nnT
2337        { \bool_if_p:N \l__enumext_store_columns_break_bool }
2338        { \bool_not_p:n { \l__enumext_starred_bool } }
2339        {
2340          \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2341        }
2342      \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.

```
2343      \bool_lazy_and:nnT
2344        { \bool_not_p:n { \l__enumext_starred_bool } }
2345        { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2346        {
2347          \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2348            {
2349              ( \exp_not:V \l__enumext_store_item_join_int )
2350            }
2351        }
```

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*.

```
2352      \bool_if:NTF \l__enumext_store_item_star_bool
2353        {
2354          \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2355          \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2356            {
2357              \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2358                {
2359                  [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2360                }
2361            }
2362          \dim_compare:nT
2363            {
2364              \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2365            }
2366            {
2367              \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2368                {
2369                  [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2370                }
2371            }
2372          \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2373        }
2374        {
2375          \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2376        }
```

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⟩.

```
2377      \bool_lazy_and:nnT
```

```
2378        { \bool_if_p:N \l__enumext_store_ref_key_bool }
2379        { \bool_if_p:N \l__enumext_hyperref_bool }
2380        {
2381          \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2382            {
2383              \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2384                    { \exp_not:V \l__enumext_mark_ref_sym_tl }
2385            }
2386        }
2387      \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2388    }
```

(*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.

```
2389  \cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
2390    {
2391      \par
2392      \bool_if:NT \l__enumext_starred_bool
2393        {
2394          \cs_set:Nn \__enumext_level: { vii }
2395        }
2396      \__enumext_print_keyans_box:cc
2397        { l__enumext_labelwidth_ \__enumext_level: _dim }
2398        { l__enumext_labelsep_ \__enumext_level: _dim }
2399      \__enumext_anskey_wrapper:n { #1 }
2400    }
```

(*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.

```
2401  \cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
2402    {
2403      \bool_if:NT \l__enumext_show_answer_bool
2404        {
2405          \__enumext_anskey_show_wrap_arg:n { #1 }
2406        }
2407      \bool_if:NT \l__enumext_show_position_bool
2408        {
2409          \tl_set:Ne \l__enumext_mark_answer_sym_tl
2410            {
2411              \group_begin:
2412              \exp_not:N \normalfont
2413              \exp_not:N \footnotesize [ \int_eval:n
2414                {
2415                  \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2416                }
2417              ]
2418              \group_end:
2419            }
2420          \__enumext_anskey_show_wrap_arg:n { #1 }
2421        }
2422    }
```

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

11.29   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*.

```
2423  \keys_define:nn { enumext / anskey }
2424    {
2425      break-col .bool_set:N = \l__enumext_store_columns_break_bool,
```

```
2426    break-col .default:n    = true,
2427    break-col .value_forbidden:n = true,
2428    item-join .int_set:N    = \l__enumext_store_item_join_int,
2429    item-join .value_required:n = true,
2430    item-star .bool_set:N = \l__enumext_store_item_star_bool,
2431    item-star .default:n    = true,
2432    item-star .value_forbidden:n = true,
2433    item-sym* .tl_set:N    = \l__enumext_store_item_symbol_tl,
2434    item-sym* .value_required:n = true,
2435    item-pos* .dim_set:N    = \l__enumext_store_item_symbol_sep_dim,
2436    item-pos* .value_required:n = true,
2437    unknown    .code:n    = { \__enumext_anskey_unknown:n {#1} },
2438  }
```

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`.

```
2439  \cs_new_protected:Npn \__enumext_anskey_unknown:n #1
2440    {
2441      \exp_args:NV \__enumext_anskey_unknown:nn \l_keys_key_str {#1}
2442    }
2443  \cs_new_protected:Npn \__enumext_anskey_unknown:nn #1 #2
2444    {
2445      \tl_if_blank:nTF {#2}
2446        {
2447          \msg_error:nnn { enumext } { anskey-cmd-key-unknown } {#1}
2448        }
2449        {
2450          \msg_error:nnnn { enumext } { anskey-cmd-key-value-unknown } {#1} {#2}
2451        }
2452    }
```

(*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`.

```
2453  \NewDocumentCommand \anskey { o +m }
2454    {
2455      \__enumext_anskey_safe_outer:
2456      \group_begin:
2457        \bool_if:NT \l__enumext_check_answers_bool
2458          {
2459            \tl_if_novalue:nF {#1}
2460              {
2461                \keys_set:nn { enumext / anskey } {#1}
2462              }
2463            \tl_if_blank:nTF {#2}
2464              {
2465                \msg_error:nn { enumext } { anskey-empty-arg }
2466              }
2467              {
2468                \__enumext_anskey_safe_inner:
2469                \__enumext_store_anskey_code:n {#2}
2470              }
2471          }
2472      \group_end:
2473    }
```

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

11.29.1 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.

```
2474  \cs_new_protected:Nn \__enumext_anskey_safe_outer:
2475    {
2476      \bool_if:NF \l__enumext_store_active_bool
```

```
2477        {
2478            \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
2479        }
2480      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2481        {
2482            \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
2483        }
2484      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2485        {
2486            \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans* }
2487        }
2488      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2489        {
2490            \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
2491        }
2492    }
```

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.

```
2493  \cs_new_protected:Nn \__enumext_anskey_safe_inner:
2494    {
2495      \int_incr:N \l__enumext_anskey_level_int
2496      \int_compare:nNnT { \l__enumext_anskey_level_int } > { 1 }
2497        {
2498            \msg_error:nn { enumext } { anskey-nested }
2499        }
2500      \bool_if:NF \l__enumext_item_number_bool
2501        {
2502            \msg_error:nn { enumext } { anskey-unnumber-item }
2503        }
2504      \mode_if_math:T
2505        {
2506            \msg_error:nne { enumext } { anskey-math-mode } { \c_backslash_str anskey }
2507        }
2508    }
```

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

11.30   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: (§11.31) which is executed after the environment in which the key save-ans is active.

```
2509  \cs_new_protected:Nn \__enumext_undefine_anskey_env:
2510    {
2511      \cs_undefine:c { anskey* }
2512      \cs_undefine:c { endanskey* }
2513      \cs_undefine:c { __scontents_anskey*_env_begin: }
2514      \cs_undefine:c { __scontents_anskey*_env_end: }
2515    }
```

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

```
2516  \__enumext_before_env:nn { enumext }
2517    {
2518      \bool_lazy_and:nnT
2519        { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2520        { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
```

```
2521        {
2522          \cs_if_free:cF { __scontents_anskey*_env_begin: }
2523            {
2524              \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2525            }
2526        }
2527      }
2528  \__enumext_before_env:nn { enumext* }
2529    {
2530      \bool_lazy_and:nnT
2531        { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2532        { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2533        {
2534          \cs_if_free:cF { __scontents_anskey*_env_begin: }
2535            {
2536              \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2537            }
2538        }
2539    }
```

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.

```
2540  \__enumext_before_env:nn { anskey* }
2541    {
2542      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2543        {
2544          \msg_error:nnn { enumext } { anskey-env-wrong }{ keyans }
2545        }
2546      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2547        {
2548          \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
2549        }
2550      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2551        {
2552          \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
2553        }
2554      \bool_if:NF \l__enumext_item_number_bool
2555        {
2556          \msg_error:nn { enumext } { anskey-unnumber-item }
2557        }
2558      \mode_if_math:T
2559        {
2560          \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
2561        }
2562    }
```

(*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: (§11.25.1) 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.

```
2563  \cs_new_protected:Npn \__enumext_anskey_env_make:n #1
2564    {
2565      \bool_if:NT \l__enumext_anskey_env_bool
2566        {
2567          \newenvsc{anskey*}[store-env=#1,print-env=false]
2568          \__enumext_anskey_env_exec:
2569        }
2570    }
2571  \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.

```
2572  \cs_new_protected:Nn \__enumext_anskey_env_define_keys:
2573    {
```

```
2574    \keys_define:nn { scontents / scontents }
2575      {
2576        break-col .bool_gset:N = \g__enumext_store_columns_break_bool,
2577        break-col .default:n   = true,
2578        break-col .value_forbidden:n = true,
2579        item-join .int_gset:N  = \g__enumxt_store_item_join_int,
2580        item-join .value_required:n = true,
2581        item-star .bool_gset:N = \g__enumext_store_item_star_bool,
2582        item-star .default:n   = true,
2583        item-star .value_forbidden:n = true,
2584        item-sym* .tl_gset:N   = \g__enumext_store_item_symbol_tl,
2585        item-sym* .value_required:n = true,
2586        item-pos* .dim_gset:N  = \g__enumext_store_item_symbol_sep_dim,
2587        item-pos* .value_required:n = true,
2588        print-env .undefine:,
2589        store-env .undefine:,
2590        write-out .undefine:,
2591        unknown   .code:n      = { \__enumext_anskey_env_unknown:n {##1} },
2592      }
2593    }
```

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.

```
2594  \cs_new_protected:Npn \__enumext_anskey_env_unknown:n #1
2595    {
2596      \exp_args:NV \__enumext_anskey_env_unknown:nn \l_keys_key_str {#1}
2597    }
2598  \cs_new_protected:Npn \__enumext_anskey_env_unknown:nn #1#2
2599    {
2600      \tl_if_blank:nTF {#2}
2601        {
2602          \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
2603        }
2604        {
2605          \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
2606        }
2607    }
```

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.

```
2608  \cs_new_protected:Nn \__enumext_anskey_env_reset_keys:
2609    {
2610      \keys_define:nn { scontents / scontents }
2611        {
2612          break-col .undefine:,
2613          item-join .undefine:,
2614          item-star .undefine:,
2615          item-sym* .undefine:,
2616          item-pos* .undefine:,
2617          write-out .code:n     = {
2618                                    \bool_set_false:N \l__scontents_storing_bool
2619                                    \bool_set_true:N  \l__scontents_writing_bool
2620                                    \tl_set:Nn \l__scontents_fname_out_tl {##1}
2621                                  },
2622          write-out .value_required:n = true,
2623          print-env .meta:nn   = { scontents } { print-env = ##1 },
2624          print-env .default:n = true,
2625          store-env .meta:nn   = { scontents } { store-env = ##1 },
2626          unknown   .code:n    = { \__scontents_parse_environment_keys:n {##1} },
2627        }
2628    }
```

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*.

```
2629  \cs_new_protected:Npn \__enumext_rescan_anskey_env:n #1
2630    {
2631      \group_begin:
2632        \int_set:Nn \tex_newlinechar:D { `\^^J }
2633        \__scontents_rescan_tokens:x
2634          {
```

```
2635          \endgroup % This assumes \catcode`\\=0... Things might go off otherwise.
2636          #1
2637        }
2638    }
```

(*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*⟩.

```
2639 \cs_new_protected:Nn \__enumext_anskey_env_exec:
2640    {
2641        \__enumext_before_env:nn { anskey* }
2642          {
2643              \__enumext_anskey_env_define_keys:
2644          }
```

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.

```
2645        \hook_if_empty:nF {env/anskey*/after}
2646          {
2647              \hook_gremove_code:nn {env/anskey*/after} { * }
2648          }
2649        \__enumext_after_env:nn { anskey* }
2650          {
2651              \__enumext_anskey_env_save_keys:
2652              \tl_clear:N \l__enumext_store_anskey_env_tl
2653              \tl_clear:N \l__enumext_store_anskey_opt_tl
2654              \bool_if:NT \l__enumext_check_answers_bool
2655                {
2656                    \tl_gset:Ne \l__enumext_store_anskey_env_tl
2657                      {
2658                          \seq_item:ce { g__scontents_name_ \l__enumext_store_name_tl _seq } { -1 }
2659                      }
2660                    \regex_match:nVTF
2661                      { ^\s* \z | ^\s* \u{c__scontents_hidden_space_str} \z }
2662                      \l__enumext_store_anskey_env_tl
2663                      {
2664                          \msg_error:nn { enumext } { anskey-empty-arg }
2665                      }
2666                      {
2667                          \__enumext_anskey_env_store:
2668                      }
2669                }
2670              \__enumext_anskey_env_clean_vars:
2671              \__enumext_anskey_env_reset_keys:
2672          }
2673    }
```

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

(*End of definition for* __enumext_anskey_env_exec:.)

__enumext_anskey_env_save_keys:
__enumext_anskey_env_store:
__enumext_anskey_env_clean_vars:

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

```
2674 \cs_new_protected:Nn \__enumext_anskey_env_save_keys:
2675    {
2676        \bool_lazy_and:nnT
2677          { \bool_if_p:N \g__enumext_store_columns_break_bool }
2678          { \bool_not_p:n { \l__enumext_starred_bool } }
2679          {
2680              \tl_put_left:Ne \l__enumext_store_anskey_opt_tl { ,break-col, }
2681          }
```

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

```
2682        \bool_lazy_and:nnT
2683          { \bool_not_p:n { \l__enumext_starred_bool } }
2684          { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }
```

```
2685        {
2686            \tl_put_left::Ne \l__enumext_store_anskey_opt_tl
2687                {
2688                    ,item-join = \exp_not:V \g__enumext_store_item_join_int,
2689                }
2690        }
```

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

```
2691        \bool_if:NT \g__enumext_store_item_star_bool
2692            {
2693            \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2694                {
2695                    ,item-star,
2696                }
2697            \tl_if_empty:NF \g__enumext_store_item_symbol_tl
2698                {
2699                \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2700                    {
2701                        ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,
2702                    }
2703                }
2704            \dim_compare:nT
2705                {
2706                    \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
2707                }
2708                {
2709                \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2710                    {
2711                        ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
2712                    }
2713                }
2714            }
2715        }
```

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`.

```
2716 \cs_new_protected:Nn \__enumext_anskey_env_store:
2717    {
2718        \group_begin:
2719        \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
2720            {
2721            \exp_args:Ne
2722                \__enumext_store_anskey_code:n
2723                    {
2724                        \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2725                    }
2726            }
2727            {
2728            \keys_set_known:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
2729            \exp_args:Ne
2730                \__enumext_store_anskey_code:n
2731                    {
2732                        \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2733                    }
2734            }
2735        \group_end:
2736    }
```

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

```
2737 \cs_new_protected:Nn \__enumext_anskey_env_clean_vars:
2738    {
2739        \bool_gset_false:N \g__enumext_store_columns_break_bool
2740        \int_gzero:N        \g__enumext_store_item_join_int
2741        \bool_gset_false:N \g__enumext_store_item_star_bool
2742        \tl_gclear:N        \g__enumext_store_item_symbol_tl
2743        \dim_gzero:N        \g__enumext_store_item_symbol_sep_dim
2744    }
```

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

11.31 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* (§11.30) 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.

```
2745  \cs_new_protected:Nn \__enumext_execute_after_env:
2746    {
2747      \int_compare:nNnT { \l__enumext_level_int } = { 0 }
2748        {
2749          \tl_if_empty:NF \g__enumext_store_name_tl
2750            {
2751              \__enumext_stop_save_ans_msg:
2752              \__enumext_item_answer_diff:
2753              \__enumext_log_global_vars:
2754              \__enumext_log_answer_vars:
2755              \bool_if:NTF \g__enumext_check_ans_key_bool
2756                {
2757                  \__enumext_check_ans_show:
2758                }
2759                { \__enumext_check_ans_log: }
2760              \__enumext_undefine_anskey_env:
2761            }
2762          \__enumext_reset_global_vars:
2763        }
2764    }
```

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

🌶 This function is passed to the function `\__enumext_after_env:nn` for the environments enumext (§11.37) and enumext* (§11.42) and it is executed only when the environments are not nested or at some level of these..

11.32 Common functions for keyans, keyans* and keyanspic

11.32.1 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`.

```
2765  \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2766    {
2767      \tl_clear:N \l__enumext_store_current_label_tl
2768      \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2769        {
2770          \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
2771        }
2772        {
2773          \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
2774        }
2775      \tl_if_novalue:nF { #1 }
2776        {
2777          % Set save-sep
2778          \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2779            {
2780              \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_o|
2781            }
2782          \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2783        }
2784      \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
2785    }
```

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

11.32.2   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.

```
2786  \cs_new_protected:Nn \__enumext_keyans_store_ref:
2787    {
2788      \bool_if:NT \l__enumext_store_ref_key_bool
2789        {
2790          \cs_set_protected:Npn \__enumext_tmp:n ##1
2791            {
2792              \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2793              \tl_reverse:c { l__enumext_label_copy_##1_tl }
2794              \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2795              \tl_reverse:c { l__enumext_label_copy_##1_tl }
2796            }
2797          \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2798          \__enumext_keyans_store_ref_aux_i:
2799        }
2800    }
```

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.

```
2801  \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2802    {
2803      \bool_if:NT \g__enumext_starred_bool
2804        {
2805          \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2806        }
2807      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2808        {
2809          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2810            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
2811        }
2812      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2813        {
2814          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2815            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2816        }
2817      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2818        {
2819          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2820            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2821        }
2822      \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2823        {
2824          \l__enumext_store_name_tl \c_colon_str
2825          \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2826        }
2827      \__enumext_keyans_store_ref_aux_ii:
2828    }
```

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.

```
2829  \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2830    {
2831      \tl_put_right:Ne \l__enumext_write_aux_file_tl
2832        {
2833          \__enumext_newlabel:nn
2834            { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2835            { \l__enumext_newlabel_arg_two_tl }
2836        }
2837      \l__enumext_write_aux_file_tl
2838    }
```

(*End of definition for* __enumext_keyans_store_ref:, __enumext_keyans_store_ref_aux_i:, *and* __enumext_keyans_store_ref_aux_ii:.)

11.32.3 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.

```
2839  \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2840    {
2841      \tl_clear:N \l__enumext_store_current_label_tl
2842      \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2843        {
2844          \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
2845        }
2846        {
2847          \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
2848        }
2849      \tl_if_novalue:nF { #1 }
2850        {
2851          \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2852            {
2853              \tl_put_right:Ne \l__enumext_store_current_label_tl
2854                {
2855                  \l__enumext_store_keyans_item_opt_sep_tl
2856                }
2857            }
2858          \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2859        }
2860      \__enumext_keyans_addto_seq_link:
2861    }
```

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.

```
2862  \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
2863    {
2864      \bool_lazy_and:nnT
2865        { \bool_if_p:N \l__enumext_store_ref_key_bool }
2866        { \bool_if_p:N \l__enumext_hyperref_bool }
2867        {
2868          \tl_put_right:Ne \l__enumext_store_current_label_tl
2869            {
2870              \hfill \exp_not:N \hyperlink
2871                {
2872                  \exp_not:V \l__enumext_newlabel_arg_one_tl
2873                }
2874                { \exp_not:V \l__enumext_mark_ref_sym_tl }
2875            }
2876        }
2877      \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
2878      \bool_if:NT \l__enumext_check_answers_bool
2879        {
2880          \int_gincr:N \g__enumext_item_anskey_int
2881        }
2882    }
```

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

11.32.4 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.

```
2883  \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
2884    {
2885      \tl_if_novalue:nF { #1 }
2886        {
```

```
2887                \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
2888              }
2889          \bool_if:NT \l__enumext_show_answer_bool
2890            {
2891                \__enumext_keyans_show_ans:
2892            }
2893          \bool_if:NT \l__enumext_show_position_bool
2894            {
2895                \__enumext_keyans_show_pos:
2896            }
2897        }
2898  \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
2899      {
2900        \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
2901          {
2902              \bool_lazy_or:nnT
2903                { \bool_if_p:N \l__enumext_show_answer_bool }
2904                { \bool_if_p:N \l__enumext_show_position_bool }
2905                {
2906                    \__enumext_keyans_wrapper_opt:n { \l__enumext_store_current_opt_arg_tl } \c_space_tl
2907                }
2908          }
2909      }
2910  \cs_new_protected:Nn \__enumext_keyans_show_ans:
2911      {
2912        \tl_put_left:Nn \l__enumext_label_v_tl
2913          {
2914              \__enumext_print_keyans_box:NN
2915                \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2916          }
2917      }
2918  \cs_new_protected:Nn \__enumext_keyans_show_pos:
2919      {
2920        \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2921          {
2922              \tl_set:Ne \l__enumext_mark_answer_sym_tl
2923                {
2924                  \group_begin:
2925                  \exp_not:N \normalfont
2926                  \exp_not:N \footnotesize [ \int_eval:n
2927                    {
2928                      \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2929                    }
2930                    ]
2931                  \group_end:
2932                }
2933          }
2934          {
2935              \tl_set:Ne \l__enumext_mark_answer_sym_tl
2936                {
2937                  \group_begin:
2938                  \exp_not:N \normalfont
2939                  \exp_not:N \footnotesize [ \int_eval:n
2940                    {
2941                      \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
2942                    }
2943                    ]
2944                  \group_end:
2945                }
2946          }
2947        \tl_put_left:Nn \l__enumext_label_v_tl
2948          {
2949              \__enumext_print_keyans_box:NN
2950                \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
2951          }
2952      }
```

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

11.33 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_ans_key_bool set by the key check-ans, set the boolean variable \l__enumext_wrap_label_X_bool to "true" and execute __enumext_item_std:w, 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.

```
2953  \cs_new_protected:Npn \__enumext_default_item:n #1
2954    {
2955      \tl_if_novalue:nTF {#1}
2956        {
2957          \bool_if:NT \l__enumext_check_answers_bool
2958            {
2959              \int_gincr:N \g__enumext_item_number_int
2960              \bool_set_true:N \l__enumext_item_number_bool
2961            }
2962          \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
2963          \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2964        }
2965        {
2966          \bool_set_eq:cc
2967            { l__enumext_wrap_label_ \__enumext_level: _bool }
2968            { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
2969          \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl
2970        }
2971    }
```

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

__enumext_starred_item:nn

The \item*, \item*[⟨*symbol*⟩] and \item*[⟨*symbol*⟩][⟨*offset*⟩] works like the numbered \item, but placing a [⟨*symbol*⟩] to the *"left"* of the ⟨*label*⟩ separated from it by the value set by the labelsep key and can be *offset* using 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 optional argument in the global variable \g__enumext_item_symbol_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_ans_key_bool set by the key check-ans, set the boolean variable \l__enumext_wrap_label_X_bool to "true" and execute __enumext_item_-std:w.

```
2972  \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
2973    {
2974      \tl_if_novalue:nF {#1}
2975        {
2976          \tl_set:cn { l__enumext_item_symbol_ \__enumext_level: _tl } {#1}
2977        }
2978      \tl_gset_eq:Nc \g__enumext_item_symbol_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
2979      \tl_if_novalue:nTF {#2}
2980        {
2981          \dim_set_eq:cc
2982            { l__enumext_item_symbol_sep_ \__enumext_level: _dim }
2983            { l__enumext_labelsep_ \__enumext_level: _dim }
2984        }
2985        {
2986          \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
2987        }
2988      \bool_if:NT \l__enumext_check_answers_bool
2989        {
2990          \int_gincr:N \g__enumext_item_number_int
2991          \bool_set_true:N \l__enumext_item_number_bool
2992        }
2993      \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
```

```
2994        \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
2995      }
```

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

__enumext_redefine_item:
__enumext_item_starred:
__enumext_make_label:

The function __enumext_redefine_item: will redefine the \item command in the enumext environment for the internal mechanism of check-answers for check-ans key and adding the starred \item* version.

```
2996  \cs_new_protected:Nn \__enumext_redefine_item:
2997    {
2998      \RenewDocumentCommand \item { s o o }
2999        {
3000          \bool_if:nTF {##1}
3001            {
3002              \__enumext_starred_item:nn {##2} {##3}
3003            }
3004            { \__enumext_default_item:n {##2} }
3005        }
3006    }
```

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

```
3007  \cs_new_protected:Nn \__enumext_item_starred:
3008    {
3009      \tl_if_empty:cF { l__enumext_item_symbol_ \__enumext_level: _tl }
3010        {
3011          \mode_leave_vertical:
3012          \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3013          \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_tl }
3014          \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3015        }
3016    }
```

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

```
3017  \cs_new_protected:Nn \__enumext_make_label:
3018    {
3019      \RenewDocumentCommand \makelabel { m }
3020        {
3021          \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
3022          \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3023          \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3024            {
3025              \__enumext_item_starred:
3026              \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3027            }
3028            { ##1 }
3029          \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
3030          \tl_gclear:N \g__enumext_item_symbol_tl
3031        }
3032    }
```

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

💣 This function is passed to __enumext_list_arg_two_X: used in the definition of the enumext environment (§11.37).

11.34 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*.

```
3033  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
3034    {
3035      \keys_define:nn { enumext / #1 }
3036        {
3037          item-sym* .tl_set:c  = { l__enumext_item_symbol_#2_tl },
3038          item-sym* .value_required:n = true,
3039          item-sym* .initial:n  = {$\star$},
3040          item-pos* .dim_set:c  = { l__enumext_item_symbol_sep_#2_dim },
3041          item-pos* .value_required:n = true,
3042        }
```

```
3043    }
3044 \clist_map_inline:nn
3045    {
3046      {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
3047    }
3048    { \__enumext_tmp:nn #1 }
```

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

11.35 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.

```
3049 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
3050    {
3051      \tl_if_novalue:nTF { #1 }
3052        {
3053          \bool_set_true:N \l__enumext_wrap_label_v_bool
3054          \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
3055        }
3056        {
3057          \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
3058          \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
3059        }
3060    }
```

(*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*⟩.

```
3061 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
3062    {
3063      \tl_set_eq:NN \l__enumext_store_current_label_tmp_tl \l__enumext_label_v_tl
3064      \__enumext_keyans_show_left:n { #1 }
3065      \bool_set_true:N \l__enumext_wrap_label_v_bool
3066      \__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*⟩.

```
3067      \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_store_current_label_tmp_tl
3068      \__enumext_keyans_addto_prop:n { #1 }
3069      \__enumext_keyans_store_ref:
3070      \__enumext_keyans_addto_seq:n { #1 }
3071      \int_gincr:N \g__enumext_check_starred_cmd_int
3072    }
```

(*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.

```
3073 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
3074    {
3075      \RenewDocumentCommand \item { s o }
3076        {
3077          \bool_if:nTF {##1}
3078            {
3079              \peek_remove_spaces:n
3080                {
3081                  \__enumext_keyans_starred_item:n {##2}
3082                }
3083            }
3084            {
3085              \__enumext_keyans_default_item:n {##2}
3086            }
```

```
3087            }
3088      }
```

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

```
3089 \cs_new_protected:Nn \__enumext_keyans_make_label:
3090   {
3091     \RenewDocumentCommand \makelabel { m }
3092       {
3093         \tl_use:N \l__enumext_label_fill_left_v_tl
3094         \tl_use:N \l__enumext_label_font_style_v_tl
3095         \bool_if:NTF \l__enumext_wrap_label_v_bool
3096           {
3097             \__enumext_wrapper_label_v:n { ##1 }
3098           }
3099           { ##1 }
3100         \tl_use:N \l__enumext_label_fill_right_v_tl
3101       }
3102   }
```

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

💣* This functions are passed to __enumext_list_arg_two_v: used in the definition of the keyans environment (§11.36.2).

11.36   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.

11.36.1   Calculation of \leftmargin and \itemindent

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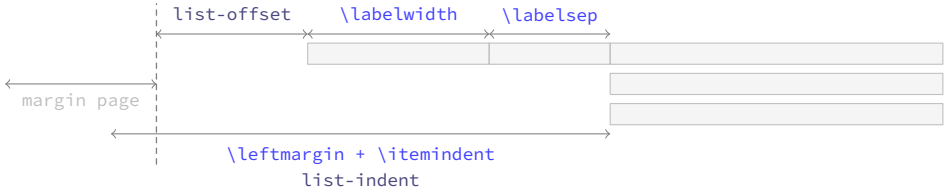
Figure 9: Representation of standard horizontal lengths in list environment.

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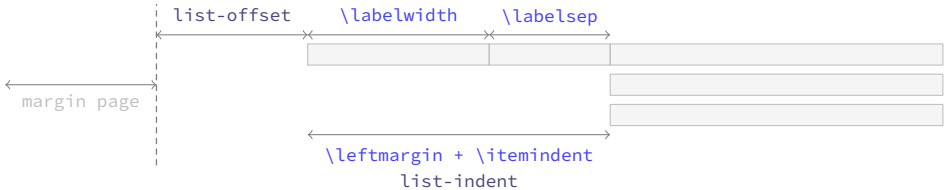
Figure 10: Representation of horizontal lengths concept in list in enumext.

Where the default values will look like in the figure 11.
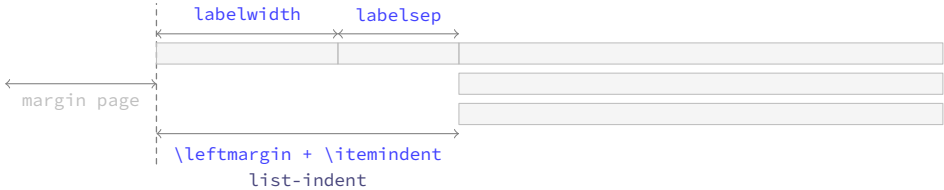
Figure 11: Default horizontal lengths in enumext.

__enumext_calc_hspace:NNNNNNN
__enumext_calc_hspace:ccccccc

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

```
    #1: \l__enumext_labelwidth_X_dim        #2: \l__enumext_labelsep_X_dim
    #3: \l__enumext_listoffset_X_dim        #4: \l__enumext_leftmargin_tmp_X_dim
```

```
                #5: \l__enumext_leftmargin_X_dim        #6: \l__enumext_itemindent_X_dim
                #7: \l__enumext_leftmargin_tmp_X_bool
```

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

This function is passed to __enumext_list_arg_two_X: which is used in the definition of the enumext and keyans environments (§11.36.2).

```
3103  \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNNN #1 #2 #3 #4 #5 #6 #7
3104    {
3105      \dim_compare:nNnT { #1 } < { \c_zero_dim }
3106        {
3107          \msg_warning:nnnV { enumext } { width-non-positive }{ labelwidth }{ #1 }
3108          \dim_set:Nn #1 { \dim_abs:n { #1 } }
3109        }
3110      \dim_compare:nNnT { #2 } < { \c_zero_dim }
3111        {
3112          \msg_warning:nnnV { enumext } { width-negative }{ labelsep }{ #2 }
3113          \dim_set:Nn #2 { \dim_abs:n { #2 } }
3114        }
```

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

```
3115        \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2} }
```

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

```
3116        \dim_compare:nNnTF { #4 } < { \c_zero_dim }
3117          {
3118            \dim_set:Nn #6 { #1 + #2 - #4}
3119            \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3120          }
3121          {
3122            \dim_compare:nNnT { #4 } = { #1 + #2 }
3123              { \dim_set:Nn #6 { \c_zero_dim } }
3124            \dim_compare:nNnT { #4 } < { #1 + #2 }
3125              { \dim_set:Nn #6 { #1 + #2 - #4} }
3126            \dim_compare:nNnT { #4 } > { #1 + #2 }
3127              {
3128                \dim_set:Nn #6 { -#1 - #2 + #4}
3129                \dim_set:Nn #6 { #6*-1}
3130              }
3131            \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3132          }
3133    }
3134  \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNNN { ccccccc }
```

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

11.36.2 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.

```
3135  \cs_set_protected:Npn \__enumext_tmp:n #1
3136    {
3137      \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3138        {
3139          \__enumext_calc_hspace:ccccccc
3140            { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3141            { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3142            { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3143            { l__enumext_leftmargin_tmp_#1_bool }
3144          \clist_map_inline:nn
3145            { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3146            { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
3147          \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3148            { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
3149          \usecounter { enumX#1 }
3150          \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
3151          \str_if_eq:nnTF {#1} { v }
3152            {
3153              \__enumext_keyans_redefine_item:
3154              \__enumext_keyans_make_label:
3155              \__enumext_keyans_ref:
```

```
3156              \__enumext_keyans_fake_item:
3157                \bool_if:cT { l__enumext_show_length_#1_bool }
3158                  {
3159                    \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3160                  }
3161              }
3162            {
3163              \__enumext_redefine_item:
3164              \__enumext_make_label:
3165              \__enumext_standar_ref:
3166              \__enumext_fake_item:
3167              \bool_if:cT { l__enumext_show_length_#1_bool }
3168                {
3169                  \msg_term:nnne { enumext } { list-lengths } {#1} { \int_use:N \l__enumext_level_i
3170                }
3171            }
3172          }
3173      }
3174  \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }
```

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

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

```
3175  \cs_set_protected:Npn \__enumext_tmp:n #1
3176    {
3177      \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3178        {
3179          \bool_set_true:c { l__enumext_leftmargin_tmp_#1_bool }
3180          \dim_zero:c { l__enumext_leftmargin_tmp_#1_dim }
3181          \__enumext_calc_hspace:ccccccc
3182            { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3183            { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3184            { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3185            { l__enumext_leftmargin_tmp_#1_bool }
3186          \clist_map_inline:nn
3187            { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3188            { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
3189          \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3190            { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
3191          \skip_set_eq:Nc \parsep { l__enumext_itemsep_#1_skip }
3192          \skip_zero:N \partopsep
3193          \usecounter { enumX#1 }
3194          \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
3195          \__enumext_starred_ref:
3196          \str_if_eq:nnTF {#1} { vii }
3197            {
3198              \__enumext_fake_item_vii:
3199              \bool_if:cT { l__enumext_show_length_vii_bool }
3200                { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3201            }
3202            {
3203              \__enumext_fake_item_viii:
3204              \bool_if:cT { l__enumext_show_length_#1_bool }
3205                { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3206            }
3207        }
3208    }
3209  \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:.)

11.37   The environment enumext

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

```
3210  \NewDocumentEnvironment{enumext}{ O{} }
3211    {
3212      \__enumext_safe_exec:
3213      \__enumext_parse_keys:n {#1}
3214      \__enumext_before_list:
```

```
3215       \__enumext_start_store_level:
3216       \__enumext_start_list:nn
3217         { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
3218         {
3219           \use:c { __enumext_list_arg_two_ \__enumext_level: : }
3220           \__enumext_before_keys_exec:
3221         }
3222       \__enumext_set_item_width:
3223       \__enumext_after_args_exec:
3224     }
3225     {
3226       \__enumext_stop_list:
3227       \__enumext_stop_store_level:
3228       \__enumext_after_list:
3229     }
```

(*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.

```
3230 \cs_new_protected:Nn \__enumext_set_item_width:
3231   {
3232     \dim_set:Nn \itemwidth
3233       {
3234         \linewidth
3235       }
3236     \dim_compare:nT
3237       {
3238         \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim } != \c_zero_dim
3239       }
3240       {
3241         \dim_sub:Nn \itemwidth
3242           {
3243             \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3244           }
3245       }
3246   }
```

(*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"*.

```
3247 \cs_new_protected:Nn \__enumext_safe_exec:
3248   {
3249     \__enumext_internal_mini_page:
3250     \__enumext_is_not_nested:
3251     \int_incr:N \l__enumext_level_int
3252     \int_compare:nNnT { \l__enumext_level_int } > { 4 }
3253       { \msg_fatal:nn { enumext } { list-too-deep } }
3254     \bool_set_true:N \l__enumext_standar_bool
3255     \__enumext_is_on_first_level:
3256   }
```

(*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.

```
3257 \cs_new_protected:Npn \__enumext_parse_keys:n #1
3258   {
3259     \tl_if_novalue:nF {#1}
3260       {
```

```
3261          \str_clear:N \l__enumext_series_str
3262          \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
3263            {
3264              \keys_set:nn { enumext / level-1 } {#1}
3265              \__enumext_parse_series:n {#1}
3266              \__enumext_nested_base_line_fix:
3267            }
3268            {
3269              \exp_args:Ne \keys_set:nn
3270                { enumext / level-\int_use:N \l__enumext_level_int } {#1}
3271            }
3272          \__enumext_store_active_keys:n {#1}
3273        }
3274      }
```

(*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*.

```
3275  \cs_new_protected:Nn \__enumext_start_store_level:
3276    {
3277      \bool_lazy_all:nT
3278        {
3279          { \bool_if_p:N \l__enumext_store_active_bool }
3280          { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3281          { \bool_if_p:N \g__enumext_standar_bool }
3282        }
3283        {
3284          \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3285            {
3286              \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3287              \__enumext_store_level_open:
3288            }
3289        }
```

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

```
3290      \bool_lazy_all:nT
3291        {
3292          { \bool_if_p:N \l__enumext_store_active_bool }
3293          { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3294          { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
3295        }
3296        {
3297          \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3298            {
3299              \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3300              \__enumext_store_level_open:
3301            }
3302        }
3303    }
```

Close the stored structure.

```
3304  \cs_new_protected:Nn \__enumext_stop_store_level:
3305    {
3306      \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
3307        {
3308          \__enumext_store_level_close:
3309        }
3310    }
```

(*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.

```
3311  \cs_new_protected:Nn \__enumext_before_list:
3312    {
3313      \__enumext_vspace_above:
3314      \__enumext_before_args_exec:
3315      \__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.

```
3316    \dim_compare:nNnT
3317       { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3318       {
3319         \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
3320           {
3321             \linewidth
3322             - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
3323             - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
3324           }
```

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*.

```
3325             \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
3326             \int_gincr:N \g__enumext_minipage_stat_int
3327             \__enumext_mini_addvspace:
3328             \nointerlineskip\noindent
3329             \begin{__enumext_mini_env*}
3330               { \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
3331           }
3332       \__enumext_multicols_start:
3333     }
```

(*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.

```
3334    \cs_new_protected:Nn \__enumext_multicols_start:
3335      {
3336        \int_compare:nNnT
3337           { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3338           {
3339             \dim_compare:nNnT
3340               { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3341               {
3342                 \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
3343                   {
3344                     ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
3345                       + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
3346                     ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
3347                     - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3348                   }
3349               }
3350             \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim  }
3351             \skip_zero:N \multicolsep
3352             \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3353               {
3354                 \dim_zero:N \columnseprule
3355               }
```

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.

```
3356             \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3357               {
3358                 \__enumext_multi_addvspace:
3359               }
3360             \raggedcolumns
```

```
3361        \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
3362      }
3363    }
```

(*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.

```
3364  \cs_new_protected:Nn \__enumext_multicols_stop:
3365    {
3366      \int_compare:nNnT
3367        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3368        {
3369          \end{multicols}
3370          \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3371            {
3372              \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
3373            }
3374        }
3375    }
```

(*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.

```
3376  \cs_new_protected:Nn \__enumext_after_list:
3377    {
3378      \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
3379        {
3380          \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3381            {
3382              \msg_warning:nn { enumext } { missing-miniright }
3383              \miniright
3384            }
3385          \int_gzero:N \g__enumext_minipage_stat_int
3386          \end{__enumext_mini_env*}
3387          \par\addvspace { \l__enumext_minipage_after_skip }
3388        }
3389        { \__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.

```
3390      \__enumext_after_stop_list:
3391      \__enumext_check_ans_key_hook:
3392      \__enumext_vspace_below:
3393      \bool_set_false:N \l__enumext_standar_bool
3394      \__enumext_resume_save_counter:
3395    }
```

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.

```
3396  \__enumext_after_env:nn {enumext} { \__enumext_execute_after_env: }
```

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

11.38   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.

```
3397  \NewDocumentEnvironment{keyans}{ O{} }
3398    {
3399      \__enumext_keyans_safe_exec:
3400      \__enumext_keyans_parse_keys:n {#1}
```

```
3401      \__enumext_before_list_v:
3402      \__enumext_start_list:nn
3403        { \tl_use:N \l__enumext_label_v_tl }
3404        {
3405          \__enumext_list_arg_two_v:
3406          \__enumext_before_keys_exec_v:
3407        }
3408      \__enumext_keyans_set_item_width:
3409      \__enumext_after_args_exec_v:
3410    }
3411    {
3412      \__enumext_check_starred_cmd:n { item }
3413      \__enumext_stop_list:
3414      \__enumext_after_list_v:
3415    }
```

(*End of definition for keyans. This function is documented on page 13.*)

__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.

```
3416  \cs_new_protected:Nn \__enumext_keyans_set_item_width:
3417    {
3418      \dim_set:Nn \itemwidth
3419        {
3420          \linewidth
3421        }
3422      \dim_compare:nT
3423        {
3424          \l__enumext_listoffset_v_dim != \c_zero_dim
3425        }
3426        {
3427          \dim_sub:Nn \itemwidth
3428            {
3429              \l__enumext_listoffset_v_dim
3430            }
3431        }
3432    }
```

(*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.

```
3433  \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3434    {
3435      \bool_if:NF \l__enumext_store_active_bool
3436        {
3437          \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
3438        }
3439      \int_incr:N \l__enumext_keyans_level_int
3440      \bool_set_true:N \l__enumext_keyans_env_bool
3441      \__enumext_keyans_start_line:
3442      % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3443      \bool_set_false:N \l__enumext_store_active_bool
3444      \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
3445        {
3446          \msg_error:nn { enumext } { keyans-nested }
3447        }
3448      \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3449        {
3450          \msg_error:nn { enumext } { keyans-wrong-level }
3451        }
3452    }
```

(*End of definition for __enumext_keyans_safe_exec:.*)

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

```
3453  \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3454    {
3455      \keys_set:nn { enumext / keyans } {#1}
3456    }
```

(End of definition for __enumext_keyans_parse_keys:n.)

Same implementation as the one used in the enumext environment.

```
3457  \cs_new_protected:Nn \__enumext_before_list_v:
3458    {
3459      \__enumext_vspace_above_v:
3460      \__enumext_before_args_exec_v:
3461      \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3462        {
3463          \dim_set:Nn \l__enumext_minipage_left_v_dim
3464            {
3465              \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3466            }
3467          \bool_set_true:N \l__enumext_minipage_active_v_bool
3468          \int_gincr:N \g__enumext_minipage_stat_int
3469          \__enumext_keyans_mini_addvspace:
3470          \nointerlineskip\noindent
3471          \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_v_dim }
3472        }
3473      \__enumext_keyans_multicols_start:
3474    }
3475  \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3476    {
3477      \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3478        {
3479          \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3480            {
3481              \dim_set:Nn \l__enumext_columns_sep_v_dim
3482                {
3483                  (
3484                    \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3485                  ) / \l__enumext_columns_v_int
3486                  - \l__enumext_listoffset_v_dim
3487                }
3488            }
3489          \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim
3490          \skip_zero:N \multicolsep
3491          \dim_zero:N \columnseprule % no rule here
3492          \bool_if:NF \l__enumext_minipage_active_v_bool
3493            {
3494              \__enumext_keyans_multi_addvspace:
3495            }
3496          \raggedcolumns
3497          \begin{multicols}{ \l__enumext_columns_v_int }
3498        }
3499    }
3500  \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3501    {
3502      \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3503        {
3504          \end{multicols}
3505          \bool_if:NF \l__enumext_minipage_active_v_bool
3506            {
3507              \par\addvspace{ \l__enumext_multicols_below_v_skip }
3508            }
3509        }
3510    }
3511  \cs_new_protected:Nn \__enumext_after_list_v:
3512    {
3513      \bool_if:NTF \l__enumext_minipage_active_v_bool
3514        {
3515          \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3516            {
3517              \msg_warning:nn { enumext } { missing-miniright }
3518              \miniright
3519            }
3520          \int_gzero:N \g__enumext_minipage_stat_int
3521          \end{__enumext_mini_env*}
3522          \par\addvspace{ \l__enumext_minipage_after_skip }
3523        }
3524        {
```

```
3525              \__enumext_keyans_multicols_stop:
3526            }
3527        \bool_set_false:N \l__enumext_keyans_env_bool
3528        \__enumext_after_stop_list_v:
3529        \__enumext_vspace_below_v:
3530      }
```

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

11.39   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.
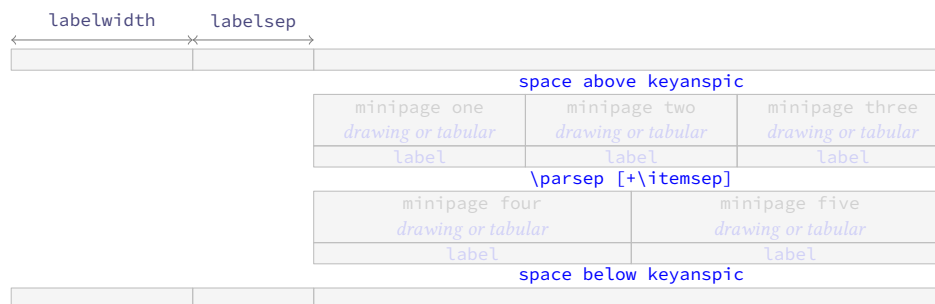
Figure 12: Representation of the keyanspic spacing in enumext.

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?.

11.39.1   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.

```
3531  \NewDocumentCommand \anspic { s o +m }
3532    {
```

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

```
3533        \bool_if:NF \l__enumext_store_active_bool
3534          {
3535            \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
3536          }
3537        \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3538          {
3539            \msg_error:nn { enumext } { keyanspic-wrong-level }
3540          }
3541        \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
3542          {
3543            \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
3544          }
```

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.

```
3545        \seq_put_right:Nn \l__enumext_keyans_pic_body_seq
3546          {
3547            \__enumext_keyans_anspic_code:nnn { #1 } { #2 } { #3 }
3548          }
3549      }
```

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

__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.

```
3550  \cs_new_protected:Nn \__enumext_keyans_anspic_code:nnn
3551    {
3552      \stepcounter { enumXvi }
```

```
3553        #3 \\
3554        \bool_if:nT { #1 }
3555          {
3556            \__enumext_keyans_addto_prop:n { #2 }
3557            \__enumext_keyans_store_ref:
3558            \__enumext_keyans_addto_seq:n { #2 }
3559            \int_gincr:N \g__enumext_check_starred_cmd_int
3560            \bool_lazy_or:nnT
3561              { \bool_if_p:N \l__enumext_show_answer_bool }
3562              { \bool_if_p:N \l__enumext_show_position_bool }
3563              {
3564                \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
3565                \__enumext_keyans_show_left:n { #2 }
3566                \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
3567              }
3568          }
3569        \tl_use:N \l__enumext_label_font_style_v_tl
3570        \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl } \__enumext_keyans_show_item_opt:
3571      }
```

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

11.39.2   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.

```
3572    \NewDocumentEnvironment{keyanspic}{ o }
3573      {
3574        \__enumext_keyans_pic_safe_exec:
3575        \__enumext_start_list:nn
3576          { }
3577          {
3578            \__enumext_keyans_pic_arg_two:
3579          }
```

We apply the "adjusted" vertical spacing above the environment

```
3580        \vspace { \l__enumext_keyans_pic_above_skip }
3581      }
```

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.

```
3582      {
3583        \tl_if_novalue:nTF { #1 }
3584          {
3585            \__enumext_keyans_pic_do:e { \seq_count:N \l__enumext_keyans_pic_body_seq }
3586          }
3587          { \__enumext_keyans_pic_do:n { #1 } }
3588        \__enumext_stop_list:
3589        \__enumext_check_starred_cmd:n { anspic }
3590        \setcounter { enumXvi } { 0 }
3591        \vspace { \l__enumext_topsep_v_skip }
3592        %\bool_set_false:N \l__enumext_store_active_bool
3593      }
```

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

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

```
3594    \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
3595      {
3596        \int_incr:N \l__enumext_keyans_pic_level_int
3597        \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
3598          {
3599            \msg_error:nn { enumext } { keyanspic-nested }
3600          }
3601        \__enumext_keyans_start_line:
3602      }
```

(*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.

```
3603 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3604   {
3605     \dim_compare:nNnT { #1 } < { 0pt }
3606       { \skip_set:Nn #1 { -#1 } }
3607   }
```

(*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.

```
3608 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
3609   {
```

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.

```
3610     \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3611     \__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.

```
3612     \skip_add:Nn \parsep { \itemsep }
3613     \dim_add:Nn  \leftmargin { -\labelwidth - \labelsep }
3614     \dim_zero:N  \labelwidth
3615     \dim_zero:N  \listparindent
3616     \dim_zero:N  \labelsep
3617     \skip_zero:N \partopsep
3618     \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.

```
3619     \__enumext_keyans_pic_skip_abs:N \parsep
3620     \skip_set:Nn \l__enumext_keyans_pic_above_skip
3621       {
3622         \box_dp:N \strutbox
3623         + \l__enumext_topsep_v_skip
3624         - \parsep
3625       }
3626     \__enumext_item_std:w \scan_stop:
3627     % paranoia
3628     \RenewDocumentCommand \item {}
3629       {
3630         \msg_error:nn { enumext } { keyanspic-item-cmd }
3631       }
3632   }
```

(*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.

```
3633 \cs_new_protected:Nn \__enumext_keyans_pic_do:n
3634   {
3635     \clist_map_function:nN { #1 } \__enumext_keyans_pic_row:n
3636   }
3637 \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.

```
3638 \cs_new_protected:Nn \__enumext_keyans_pic_row:n
3639   {
3640     \dim_set:Nn \l__enumext_keyans_pic_width_dim { \linewidth / #1 }
3641     \int_set:Nn \l__enumext_keyans_pic_above_int { \l__enumext_keyans_pic_below_int }
3642     \int_set:Nn \l__enumext_keyans_pic_below_int { \l__enumext_keyans_pic_above_int + #1 }
3643     \int_step_inline:nnn
```

```
3644      { \l__enumext_keyans_pic_above_int + 1 }
3645      { \l__enumext_keyans_pic_below_int }
3646      {
3647        \__enumext_minipage:w [ b ]{ \l__enumext_keyans_pic_width_dim }
3648          \centering
3649          \seq_item:Nn \l__enumext_keyans_pic_body_seq { ##1 }
3650        \__enumext_endminipage:
3651      }
3652    \par
3653  }
```

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

11.40 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.

11.41 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.

```
3654  \cs_new_protected:Nn \__enumext_footnotetext:nn
3655    {
3656      \footnotetext[#1]{#2}
3657    }
3658  \cs_new_protected:Nn \__enumext_renew_footnote:
3659    {
3660      \seq_gclear:N \g__enumext_footnote_arg_seq
3661      \seq_gclear:N \g__enumext_footnote_int_seq
3662      \RenewDocumentCommand \footnote { o +m }
3663        {
3664          \tl_if_novalue:nTF {##1}
3665            {
3666              \stepcounter{footnote}
3667              \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
3668            }
3669            {
3670              \int_gset:Nn \g__enumext_footnote_int { ##1 }
3671            }
3672          \footnotemark [ \g__enumext_footnote_int ]
3673          \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
3674          \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
3675        }
3676    }
3677  \cs_new_protected:Nn \__enumext_print_footnote:
3678    {
3679      \seq_if_empty:NF \g__enumext_footnote_int_seq
3680        {
3681          \seq_map_pairwise_function:NNN
3682            \g__enumext_footnote_int_seq
3683            \g__enumext_footnote_arg_seq
3684            \__enumext_footnotetext:nn
3685        }
3686    }
```

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

11.41.1 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.

```
3687  \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
```

```
3688    {
3689      \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
3690        {
3691          \dim_set:Nn \l__enumext_columns_sep_vii_dim
3692            {
3693              ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
3694              / \l__enumext_columns_vii_int
3695            }
3696        }
3697      \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - 1 }
3698      \dim_set:Nn \l__enumext_item_width_vii_dim
3699        {
3700          ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
3701          / \l__enumext_columns_vii_int
3702          - \l__enumext_labelwidth_vii_dim
3703          - \l__enumext_labelsep_vii_dim
3704        }
```

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

```
3705        \dim_compare:nNnT { \l__enumext_rightmargin_vii_dim } > { \c_zero_dim }
3706          {
3707            \dim_sub:Nn \l__enumext_item_width_vii_dim
3708              {
3709                ( \l__enumext_rightmargin_vii_dim * \l__enumext_tmpa_vii_int )
3710                / \l__enumext_columns_vii_int
3711              }
3712            \dim_add:Nn \l__enumext_columns_sep_vii_dim
3713              {
3714                \l__enumext_rightmargin_vii_dim
3715              }
3716          }
3717      }
```

Same implementation for the `keyans*` environment.

```
3718  \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
3719    {
3720      \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
3721        {
3722          \dim_set:Nn \l__enumext_columns_sep_viii_dim
3723            {
3724              ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
3725              / \l__enumext_columns_viii_int
3726            }
3727        }
3728      \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - 1 }
3729      \dim_set:Nn \l__enumext_item_width_viii_dim
3730        {
3731          ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
3732          / \l__enumext_columns_viii_int
3733          - \l__enumext_labelwidth_viii_dim
3734          - \l__enumext_labelsep_viii_dim
3735        }
3736      \dim_compare:nNnT { \l__enumext_rightmargin_viii_dim } > { \c_zero_dim }
3737        {
3738          \dim_sub:Nn \l__enumext_item_width_viii_dim
3739            {
3740              ( \l__enumext_rightmargin_viii_dim * \l__enumext_tmpa_vii_int )
3741              / \l__enumext_columns_viii_int
3742            }
3743          \dim_add:Nn \l__enumext_columns_sep_viii_dim
3744            {
3745              \l__enumext_rightmargin_viii_dim
3746            }
3747        }
3748    }
```

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

11.41.2   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.

```
3749  \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
3750    {
3751      \int_set:Nn \l__enumext_joined_item_vii_int {#1}
3752      \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
3753        {
3754          \msg_warning:nnee { enumext } { item-joined }
3755            { \int_use:N \l__enumext_joined_item_vii_int }
3756            { \int_use:N \l__enumext_columns_vii_int }
3757          \int_set:Nn \l__enumext_joined_item_vii_int
3758            {
3759              \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
3760            }
3761        }
3762      \int_compare:nNnT
3763        { \l__enumext_joined_item_vii_int }
3764          >
3765        { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
3766        {
3767          \msg_warning:nnee { enumext } { item-joined-columns }
3768            { \int_use:N \l__enumext_joined_item_vii_int }
3769            {
3770              \int_eval:n
3771                { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
3772            }
3773          \int_set:Nn \l__enumext_joined_item_vii_int
3774            {
3775              \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
3776            }
3777        }
3778      \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { 1 }
3779        {
3780          \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
3781          \int_decr:N \l__enumext_joined_item_aux_vii_int
3782          \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
3783          \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
3784          \dim_set:Nn \l__enumext_joined_width_vii_dim
3785            {
3786              \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
3787              + (  \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
3788                + \l__enumext_columns_sep_vii_dim
3789              )*\l__enumext_joined_item_aux_vii_int
3790            }
3791          \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
3792        }
3793        {
3794          \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
3795          \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
3796        }
3797    }
```

Same implementation for the keyans* environment.

```
3798  \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
3799    {
3800      \int_set:Nn \l__enumext_joined_item_viii_int {#1}
3801      \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
3802        {
3803          \msg_warning:nnee { enumext } { item-joined }
3804            { \int_use:N \l__enumext_joined_item_viii_int }
3805            { \int_use:N \l__enumext_columns_viii_int }
3806          \int_set:Nn \l__enumext_joined_item_viii_int
3807            {
3808              \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
3809            }
3810        }
3811      \int_compare:nNnT
3812        { \l__enumext_joined_item_viii_int }
3813          >
3814        { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
3815        {
3816          \msg_warning:nnee { enumext } { item-joined-columns }
3817            { \int_use:N \l__enumext_joined_item_viii_int }
3818            {
```

```
3819        \int_eval:n
3820          { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
3821        }
3822      \int_set:Nn \l__enumext_joined_item_viii_int
3823        {
3824          \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
3825        }
3826      }
3827    \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { 1 }
3828      {
3829        \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
3830        \int_decr:N \l__enumext_joined_item_aux_viii_int
3831        \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
3832        \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
3833        \dim_set:Nn \l__enumext_joined_width_viii_dim
3834          {
3835            \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
3836            + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
3837                + \l__enumext_columns_sep_viii_dim
3838              )*\l__enumext_joined_item_aux_viii_int
3839          }
3840        \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
3841      }
3842      {
3843        \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
3844        \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
3845      }
3846    }
```

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

11.41.3 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.

```
3847  \cs_new_protected:Nn \__enumext_start_mini_vii:
3848    {
3849      \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
3850        {
3851          \dim_set:Nn \l__enumext_minipage_left_vii_dim
3852            {
3853              \linewidth
3854              - \l__enumext_minipage_right_vii_dim
3855              - \l__enumext_minipage_hsep_vii_dim
3856            }
3857          \bool_set_true:N \l__enumext_minipage_active_vii_bool
3858          \dim_gset_eq:NN
3859            \g__enumext_minipage_right_vii_dim
3860            \l__enumext_minipage_right_vii_dim
3861          \__enumext_mini_addvspace_vii:
3862          \nointerlineskip\noindent
3863          \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_vii_dim }
3864        }
3865    }
```

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"*.

```
3866  \cs_new_protected:Nn \__enumext_stop_mini_vii:
3867    {
3868      \bool_if:NT \l__enumext_minipage_active_vii_bool
3869        {
3870          \end{__enumext_mini_env*}
3871          \hfill
3872          \bool_gset_true:N \g__enumext_minipage_active_vii_bool
3873        }
3874    }
```

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

```
3875  \__enumext_after_env:nn {enumext*}
3876    {
3877      \bool_if:NT \g__enumext_minipage_active_vii_bool
3878        {
3879          \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_vii_dim }
3880            \par\addvspace { \g__enumext_minipage_right_skip }
3881            \bool_if:NF \g__enumext_minipage_center_vii_bool
3882              {
3883                \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
3884                  {
3885                    \centering
3886                  }
3887              }
3888            \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
3889              {
3890                \tl_use:N \g__enumext_miniright_code_vii_tl
3891              }
3892            \box_use_drop:N \l__enumext_miniright_code_vii_box
3893          \end{__enumext_mini_env*}
3894          \par\addvspace{ \g__enumext_minipage_after_skip }
3895        }
3896      \bool_gset_false:N \g__enumext_minipage_active_vii_bool
3897      \bool_gset_true:N \g__enumext_minipage_center_vii_bool
3898      \tl_gclear:N \g__enumext_miniright_code_vii_tl
3899      \dim_gzero:N \g__enumext_minipage_right_vii_dim
3900      \bool_gset_false:N \g__enumext_starred_bool
3901    }
```

(*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.

```
3902  \cs_new_protected:Nn \__enumext_start_mini_viii:
3903    {
3904      \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
3905        {
3906          \dim_set:Nn \l__enumext_minipage_left_viii_dim
3907            {
3908              \linewidth
3909              - \l__enumext_minipage_right_viii_dim
3910              - \l__enumext_minipage_hsep_viii_dim
3911            }
3912          \bool_set_true:N \l__enumext_minipage_active_viii_bool
3913          \dim_gset_eq:NN
3914            \g__enumext_minipage_right_viii_dim
3915            \l__enumext_minipage_right_viii_dim
3916          \__enumext_mini_addvspace_viii:
3917          \nointerlineskip\noindent
3918          \begin{__enumext_mini_env*}{ \l__enumext_minipage_left_viii_dim }
3919        }
3920    }
3921  \cs_new_protected:Nn \__enumext_stop_mini_viii:
3922    {
3923      \bool_if:NT \l__enumext_minipage_active_viii_bool
3924        {
3925          \end{__enumext_mini_env*}
3926          \hfill
3927          \bool_gset_true:N \g__enumext_minipage_active_viii_bool
3928        }
3929    }
3930  \__enumext_after_env:nn {keyans*}
3931    {
3932      \bool_if:NT \g__enumext_minipage_active_viii_bool
3933        {
3934          \begin{__enumext_mini_env*}{ \g__enumext_minipage_right_viii_dim }
3935            \par\addvspace { \g__enumext_minipage_right_skip }
3936            \bool_if:NF \g__enumext_minipage_center_viii_bool
```

```
3937              {
3938                \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
3939                  {
3940                    \centering
3941                  }
3942              }
3943            \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
3944              {
3945                \tl_use:N \g__enumext_miniright_code_viii_tl
3946              }
3947            \box_use_drop:N \l__enumext_miniright_code_viii_box
3948          \end{__enumext_mini_env*}
3949          \par\addvspace{ \g__enumext_minipage_after_skip }
3950        }
3951      \bool_gset_false:N \g__enumext_minipage_active_viii_bool
3952      \bool_gset_true:N \g__enumext_minipage_center_viii_bool
3953      \tl_gclear:N \g__enumext_miniright_code_viii_tl
3954      \dim_gzero:N \g__enumext_minipage_right_viii_dim
3955    }
```

(*End of definition for* __enumext_start_mini_viii: *and* __enumext_stop_mini_viii:*.*)

11.42 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.

```
3956  \NewDocumentEnvironment{enumext*}{ o }
3957    {
3958      \__enumext_safe_exec_vii:
3959      \__enumext_parse_keys_vii:n {#1}
3960      \__enumext_before_list_vii:
3961      \__enumext_start_store_level_vii:
3962      \__enumext_start_list:nn { }
3963        {
3964          \__enumext_list_arg_two_vii:
3965          \__enumext_before_keys_exec_vii:
3966        }
3967      \__enumext_starred_columns_set_vii:
3968      \item[] \scan_stop:
3969      \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \noindent
3970      \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
3971    }
3972    {
3973      \__enumext_stop_item_tmp_vii:
3974      \__enumext_remove_extra_parsep_vii:
3975      \__enumext_stop_list:
3976      \__enumext_stop_store_level_vii:
3977      \__enumext_after_list_vii:
3978    }
```

(*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.

```
3979  \cs_new_protected:Nn \__enumext_safe_exec_vii:
3980    {
3981      \__enumext_internal_mini_page:
3982      \__enumext_is_not_nested:
3983      \int_incr:N \l__enumext_level_h_int
3984      \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
3985        {
3986          \msg_error:nn { enumext } { nested }
3987        }
3988      \bool_set_true:N \l__enumext_starred_bool
3989      \__enumext_is_on_first_level:
3990    }
```

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

__enumext_parse_keys_vii:n First we will clear the variable \l__enumext_series_str used by the key series, process the environment [⟨*key* = *val*⟩] and execute the function __enumext_parse_series:n and used by the key series, then we execute the function __enumext_store_active_keys_vii:n and reprocess the ⟨*keys*⟩ to pass them to the storage ⟨*sequence*⟩ if the key save-key is not active and finally we call the function __enumext_nested_base_line_fix: used by the key base-fix.

```
3991 \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
3992   {
3993     \tl_if_novalue:nF {#1}
3994       {
3995         \str_clear:N \l__enumext_series_str
3996         \keys_set:nn { enumext / enumext* } {#1}
3997         \__enumext_parse_series:n {#1}
3998         \__enumext_store_active_keys_vii:n {#1}
3999         \__enumext_nested_base_line_fix:
4000       }
4001   }
```

(*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*.

```
4002 \cs_new_protected:Nn \__enumext_before_list_vii:
4003   {
4004     \__enumext_vspace_above_vii:
4005     \__enumext_check_ans_active:
4006     \__enumext_before_args_exec_vii:
4007     \__enumext_start_mini_vii:
4008   }
```

(*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.

```
4009 \cs_new_protected:Nn \__enumext_after_list_vii:
4010   {
4011     \__enumext_stop_mini_vii:
4012     \__enumext_after_stop_list_vii:
4013     \__enumext_check_ans_key_hook:
4014     \__enumext_vspace_below_vii:
4015     \bool_set_false:N \l__enumext_starred_bool
4016     \__enumext_resume_save_counter:
4017   }
```

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

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

```
4018 \cs_new_protected:Nn \__enumext_start_store_level_vii:
4019   {
4020     \bool_if:NT \l__enumext_store_active_bool
4021       {
4022         \int_compare:nNnT { \l__enumext_level_int } > { 0 }
4023           {
4024             \__enumext_store_level_open_vii:
4025           }
4026       }
4027   }
4028 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
4029   {
4030     \bool_if:NT \l__enumext_store_active_bool
4031       {
```

```
4032         \int_compare:nNnT { \l__enumext_level_int } > { 0 }
4033           {
4034             \__enumext_store_level_close_vii:
4035           }
4036       }
4037   }
```

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

11.42.1 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.

```
4038 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
4039   {
4040     \__enumext_stop_item_tmp_vii:
4041     \int_incr:N \l__enumext_item_column_pos_vii_int
4042     \int_gincr:N \g__enumext_item_count_all_vii_int
4043     \__enumext_item_peek_args_vii:
4044   }
```

(*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).

```
4045 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
4046   {
4047     \peek_meaning:NTF (
4048       { \__enumext_joined_item_vii:w }
4049       { \__enumext_joined_item_vii:w (1) }
4050   }
```

(*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.

```
4051 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
4052   {
4053     \__enumext_starred_joined_item_vii:n {#1}
4054     \peek_meaning_remove:NTF *
4055       { \__enumext_starred_item_vii:w  }
4056       { \__enumext_standar_item_vii:w }
4057   }
```

(*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].

```
4058 \cs_new_protected:Npn \__enumext_standar_item_vii:w
4059   {
4060     \bool_set_false:N \l__enumext_item_starred_vii_bool
4061       \peek_meaning:NTF [
4062         {
4063           \bool_set_eq:NN
4064             \l__enumext_wrap_label_vii_bool
4065             \l__enumext_wrap_label_opt_vii_bool
4066             \__enumext_start_item_vii:w
4067         }
```

```
4068              {
4069                  \bool_set_true:N \l__enumext_wrap_label_vii_bool
4070                  \legacy_if_set_true:n { @noitemarg }
4071                  \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4072              }
4073          }
```

(*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*⟩].

```
4074  \cs_new_protected:Npn \__enumext_starred_item_vii:w
4075    {
4076      \bool_set_true:N \l__enumext_item_starred_vii_bool
4077      \bool_set_true:N \l__enumext_wrap_label_vii_bool
4078      \peek_meaning:NTF [
4079        { \__enumext_starred_item_vii_aux_i:w }
4080        { \__enumext_starred_item_vii_aux_ii:w }
4081    }
4082  \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
4083    {
4084      \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
4085      \__enumext_starred_item_vii_aux_ii:w
4086    }
4087  \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
4088    {
4089      \peek_meaning:NTF [
4090        { \__enumext_starred_item_vii_aux_iii:w }
4091        {
4092          \dim_set_eq:NN
4093            \l__enumext_item_symbol_sep_vii_dim
4094            \l__enumext_labelsep_vii_dim
4095          \legacy_if_set_true:n { @noitemarg }
4096          \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4097        }
4098    }
4099  \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
4100    {
4101      \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
4102      \legacy_if_set_true:n { @noitemarg }
4103      \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4104    }
```

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

11.42.2   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.

```
4105  \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
4106    {
4107      \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
4108      \legacy_if:nT { @noitemarg }
4109        {
4110          \legacy_if_set_false:n { @noitemarg }
4111          \legacy_if:nT { @nmbrlist }
4112            {
4113              \bool_if:NT \l__enumext_hyperref_bool
4114                {
4115                  \legacy_if_set_true:n { @hyper@item }
4116                }
4117              \refstepcounter{enumXvii}
4118              \bool_if:NT \l__enumext_check_answers_bool
4119                {
4120                  \int_gincr:N \g__enumext_item_number_int
4121                  \bool_set_true:N \l__enumext_item_number_bool
```

```
4122                       }
4123                   }
4124               }
```

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.

```
4125         \group_begin:
4126           \lrbox{ \l__enumext_item_text_vii_box }
4127             \bool_if:NF \l__enumext_footnotes_key_bool
4128               {
4129                  \__enumext_renew_footnote:
4130               }
4131             \bool_if:NT \l__enumext_item_starred_vii_bool
4132               {
4133                  \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
4134                    {
4135                       \tl_gset_eq:NN
4136                          \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
4137                    }
4138                  \mode_leave_vertical:
4139                  \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
4140                  \makebox[ 0pt ][ r ]{ \g__enumext_item_symbol_aux_vii_tl }
4141                  \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
4142                  \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
4143               }
4144             \group_begin:
4145               \tl_use:N \l__enumext_label_font_style_vii_tl
4146               \bool_if:NTF \l__enumext_wrap_label_vii_bool
4147                 {
4148                    \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
4149                      { \__enumext_wrapper_label_vii:n {#1} }
4150                 }
4151                 {
4152                    \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]{ #1 }
4153                 }
4154             \group_end:
4155             \skip_horizontal:N \l__enumext_labelsep_vii_dim
4156             \tl_use:N \l__enumext_after_list_args_vii_tl
4157             \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim  }
4158               \skip_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
4159               \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
4160               \tl_use:N \l__enumext_fake_item_indent_vii_tl
4161           }
```

(*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.

```
4162 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
4163   {
4164         \__enumext_endminipage:
4165       \endlrbox
4166     \group_end:
4167     \box_set_wd:Nn \l__enumext_item_text_vii_box
4168       {
4169          \l__enumext_joined_width_vii_dim
4170        + \l__enumext_labelwidth_vii_dim
4171        + \l__enumext_labelsep_vii_dim
4172       }
4173     \int_set:Nn \hbadness { 10000 }
4174     \box_use_drop:N \l__enumext_item_text_vii_box
4175     \bool_if:NF \l__enumext_footnotes_key_bool
4176       {
4177          \__enumext_print_footnote:
4178       }
```

```
4179    \int_compare:nNnTF { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
4180      {
4181        \par\noindent
4182        \int_zero:N \l__enumext_item_column_pos_vii_int
4183      }
4184      { \hspace{ \l__enumext_columns_sep_vii_dim } }
4185    }
```

(*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.

```
4186  \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
4187    {
4188      \int_compare:nNnT
4189        {
4190          \int_mod:nn {  \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
4191        }
4192        =
4193        { 0 }
4194        {
4195          \par
4196          \vspace{ -\l__enumext_itemsep_vii_skip }
4197          \int_gzero:N \g__enumext_item_count_all_vii_int
4198        }
4199    }
```

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.

```
4200  \__enumext_after_env:nn {enumext*} { \__enumext_execute_after_env: }
```

(*End of definition for* __enumext_remove_extra_parsep_vii:.)

11.43   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.

```
4201  \NewDocumentEnvironment{keyans*}{ o }
4202    {
4203      \__enumext_safe_exec_viii:
4204      \__enumext_parse_keys_viii:n {#1}
4205      \__enumext_before_list_viii:
4206      \__enumext_start_list:nn { }
4207        {
4208          \__enumext_list_arg_two_viii:
4209          \__enumext_before_keys_exec_viii:
4210        }
4211      \__enumext_starred_columns_set_viii:
4212      \item[] \scan_stop:
4213      \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \noindent
4214      \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
4215    }
4216    {
4217      \__enumext_stop_item_tmp_viii:
4218      \__enumext_remove_extra_parsep_viii:
4219      \__enumext_check_starred_cmd:n { item }
4220      \__enumext_stop_list:
4221      \__enumext_after_list_viii:
4222    }
```

(*End of definition for* keyans*. *This function is documented on page 13.*)

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

```
4223  \cs_new_protected:Nn \__enumext_safe_exec_viii:
4224    {
4225      \int_incr:N \l__enumext_keyans_level_h_int
4226      \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
4227        {
4228          \msg_error:nn { enumext } { nested }
4229        }
```

```
4230     \__enumext_keyans_start_line:
4231     % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
4232     \bool_set_false:N \l__enumext_store_active_bool
4233     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
4234       {
4235         \msg_error:nn { enumext } { keyans-wrong-level }
4236       }
4237   }
```

(*End of definition for* __enumext_safe_exec_viii:.)

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

```
4238 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
4239   {
4240     \tl_if_novalue:nF {#1}
4241       {
4242         \keys_set:nn { enumext / keyans* } {#1}
4243       }
4244   }
```

(*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.

```
4245 \cs_new_protected:Nn \__enumext_before_list_viii:
4246   {
4247     \__enumext_vspace_above_viii:
4248     \__enumext_before_args_exec_viii:
4249     \__enumext_start_mini_viii:
4250   }
```

(*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.

```
4251 \cs_new_protected:Nn \__enumext_after_list_viii:
4252   {
4253     \__enumext_stop_mini_viii:
4254     \__enumext_after_stop_list_viii:
4255     \__enumext_vspace_below_viii:
4256   }
```

(*End of definition for* __enumext_after_list_viii:.)

11.43.1 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.

```
4257 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
4258   {
4259     \__enumext_stop_item_tmp_viii:
4260     \int_incr:N \l__enumext_item_column_pos_viii_int
4261     \int_gincr:N \g__enumext_item_count_all_viii_int
4262     \__enumext_item_peek_args_viii:
4263   }
```

(*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).

```
4264 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
4265   {
4266     \peek_meaning:NTF (
4267       { \__enumext_joined_item_viii:w }
4268       { \__enumext_joined_item_viii:w (1) }
4269   }
```

(*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.

```
4270 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
4271   {
4272     \__enumext_starred_joined_item_viii:n {#1}
4273     \peek_meaning_remove:NTF *
4274       { \__enumext_starred_item_viii:w  }
4275       { \__enumext_standar_item_viii:w }
4276   }
```

(*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].

```
4277 \cs_new_protected:Npn \__enumext_standar_item_viii:w
4278   {
4279     \bool_set_false:N \l__enumext_item_starred_viii_bool
4280       \peek_meaning:NTF [
4281         {
4282           \bool_set_eq:NN
4283             \l__enumext_wrap_label_viii_bool
4284             \l__enumext_wrap_label_opt_viii_bool
4285           \__enumext_start_item_viii:w
4286         }
4287         {
4288           \bool_set_true:N \l__enumext_wrap_label_viii_bool
4289           \legacy_if_set_true:n { @noitemarg }
4290           \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4291         }
4292   }
```

(*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*⟩].

```
4293 \cs_new_protected:Npn \__enumext_starred_item_viii:w
4294   {
4295     \bool_set_true:N \l__enumext_item_starred_viii_bool
4296     \bool_set_true:N \l__enumext_wrap_label_viii_bool
4297     \peek_meaning:NTF [
4298       { \__enumext_starred_item_viii_aux_i:w }
4299       { \__enumext_starred_item_viii_aux_ii:w }
4300   }
```

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.

```
4301 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
```

```
4302    {
4303      \tl_clear:N \l__enumext_store_current_label_tl
4304      \tl_if_novalue:nF { #1 }
4305        {
4306          \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
4307            {
4308              \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_op
4309              \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
4310            }
4311          \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
4312        }
4313      \__enumext_starred_item_viii_aux_ii:w
4314    }
4315 \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
4316    {
4317      \legacy_if_set_true:n { @noitemarg }
4318      \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4319    }
```

(*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.

```
4320 \cs_new_protected:Nn \__enumext_starred_item_exec:
4321    {
4322      \tl_put_left:Ne \l__enumext_store_current_label_tl { \l__enumext_label_viii_tl }
4323      \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
4324      \__enumext_keyans_store_ref:
4325      \tl_put_left:Ne \l__enumext_store_current_label_tl { \item }
4326      \__enumext_keyans_addto_seq_link:
4327      \int_gincr:N \g__enumext_check_starred_cmd_int
4328      \bool_if:NT \l__enumext_show_answer_bool
4329        {
4330          \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4331        }
4332      \bool_if:NT \l__enumext_show_position_bool
4333        {
4334          \tl_set:Ne \l__enumext_mark_answer_sym_tl
4335            {
4336              \group_begin:
4337                \exp_not:N \normalfont
4338                \exp_not:N \footnotesize [ \int_eval:n
4339                  {
4340                    \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
4341                  }
4342                ]
4343              \group_end:
4344            }
4345          \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4346        }
4347    }
```

(*End of definition for* __enumext_starred_item_exec:.)

11.43.2 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.

```
4348 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
4349    {
4350      \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
4351      \legacy_if:nT { @noitemarg }
4352        {
4353          \legacy_if_set_false:n { @noitemarg }
4354          \legacy_if:nT { @nmbrlist }
4355            {
4356              \bool_if:NT \l__enumext_hyperref_bool
4357                {
4358                  \legacy_if_set_true:n { @hyper@item }
4359                }
```

```
4360                  \refstepcounter{enumXviii}
4361              }
4362          }
```

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

```
4363      \group_begin:
4364        \lrbox{ \l__enumext_item_text_viii_box }
4365          \bool_if:NF \l__enumext_footnotes_key_bool
4366            {
4367                \__enumext_renew_footnote:
4368            }
4369          \bool_if:NT \l__enumext_item_starred_viii_bool
4370            {
4371                \__enumext_starred_item_exec:
4372            }
4373          \group_begin:
4374            \tl_use:N \l__enumext_label_font_style_viii_tl
4375            \bool_if:NTF \l__enumext_wrap_label_viii_bool
4376              {
4377                  \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
4378                    { \__enumext_wrapper_label_viii:n {#1} }
4379              }
4380              {
4381                  \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]{ #1
4382              }
4383          \group_end:
4384          \skip_horizontal:N \l__enumext_labelsep_viii_dim
4385          \tl_use:N \l__enumext_after_list_args_viii_tl
4386          \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim  }
4387            \skip_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
4388            \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
4389            \bool_if:NT \l__enumext_item_starred_viii_bool
4390              {
4391                  \tl_use:N \l__enumext_fake_item_indent_viii_tl
4392                  \__enumext_keyans_show_item_opt:
4393                  \skip_horizontal:n { -\l__enumext_fake_item_indent_viii_dim - \l__enumext_labelsep_v
4394              }
4395              {
4396                  \tl_use:N \l__enumext_fake_item_indent_viii_tl
4397              }
4398      }
```

(*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.

```
4399  \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
4400    {
4401          \__enumext_endminipage:
4402        \endlrbox
4403      \group_end:
4404      \box_set_wd:Nn \l__enumext_item_text_viii_box
4405        {
4406          \l__enumext_joined_width_viii_dim
4407          + \l__enumext_labelwidth_viii_dim
4408          + \l__enumext_labelsep_viii_dim
4409        }
4410      \int_set:Nn \hbadness { 10000 }
4411      \box_use_drop:N \l__enumext_item_text_viii_box
4412      \bool_if:NF \l__enumext_footnotes_key_bool
4413        {
4414          \__enumext_print_footnote:
4415        }
4416      \int_compare:nNnTF
4417        { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
4418        {
4419          \par\noindent
4420          \int_zero:N \l__enumext_item_column_pos_viii_int
4421        }
4422        { \hspace{ \l__enumext_columns_sep_viii_dim } }
4423    }
```

(*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.

```
4424  \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
4425    {
4426      \int_compare:nNnT
4427        {
4428          \int_mod:nn
4429            { \g__enumext_item_count_all_viii_int }
4430            { \l__enumext_columns_viii_int }
4431        }
4432      =
4433      { 0 }
4434      {
4435        \par
4436        \vspace{ -\l__enumext_itemsep_viii_skip }
4437        \int_gzero:N \g__enumext_item_count_all_viii_int
4438      }
4439    }
```

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

11.44 The command `\getkeyans`

`\getkeyans` The `\getkeyans` command takes a mandatory argument of the form $\{\langle$*store name : position*$\rangle\}$. Retrieve a "*single*" content stored by `\anskey`, `\anspic*` and `\item*` from $\langle$*prop list*$\rangle$ defined by `save-ans` key.

```
4440  \NewDocumentCommand \getkeyans { m }
4441    {
4442      \exp_args:Ne \__enumext_getkeyans_aux:n
4443        { \tl_to_str:e { \text_expand:n {#1} } }
4444    }
```

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

`\__enumext_getkeyans_aux:n` The internal function `\__enumext_getkeyans_aux:n` is in charge of *splitting* the $\langle$*argument*$\rangle$ using ":". If ":" is omitted it will return an error.

```
4445  \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
4446    {
4447      \str_if_in:nnTF {#1} { : }
4448        {
4449          \use:e
4450            {
4451              \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
4452                { {##1} {##2} }
4453            }
4454          \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
4455        }
4456        { \msg_error:nnn { enumext } { missing-colon } {#1} }
4457    }
```

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

`\__enumext_getkeyans:nn` The internal function `\__enumext_getkeyans:nn` will check for the existence of the $\langle$*prop list*$\rangle$, if it does not exist it will return an error message, then it will fetch the content specified by the second $\langle$*argument*$\rangle$ from $\langle$*prop list*$\rangle$.

```
4458  \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
4459    {
4460      \prop_if_exist:cF { g__enumext_#1_prop }
4461        { \msg_error:nnn { enumext } { undefined-storage-anskey } {#1} }
4462      \group_begin:
4463        \prop_item:cn { g__enumext_#1_prop }{#2}
4464      \group_end:
4465    }
```

(*End of definition for* `\__enumext_getkeyans:nn`*.*)

11.45 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*⟩]

```
4466 \cs_generate_variant:Nn \keys_precompile:nnN { neN }
4467 \keys_define:nn  { enumext / print }
4468   {
4469     print*  .code:n     = \keys_precompile:neN { enumext / enumext* }
4470                             { \__enumext_filter_save_key:n {#1} }
4471                             \l__enumext_print_keyans_starred_tl, % starred cmd
4472     print*  .initial:n  = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4473     print-1 .code:n     = \keys_precompile:neN { enumext / level-1 }
4474                             { \__enumext_filter_save_key:n {#1} }
4475                             \l__enumext_print_keyans_i_tl,
4476     print-1 .initial:n  = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
4477     print-2 .code:n     = \keys_precompile:neN { enumext / level-2 }
4478                             { \__enumext_filter_save_key:n {#1} }
4479                             \l__enumext_print_keyans_ii_tl,
4480     print-2 .initial:n  = { nosep, label=(\alph*), first=\small, font=\small },
4481     print-3 .code:n     = \keys_precompile:neN { enumext / level-3 }
4482                             { \__enumext_filter_save_key:n {#1} }
4483                             \l__enumext_print_keyans_iii_tl,
4484     print-3 .initial:n  = { nosep, label=\roman*., first=\small, font=\small },
4485     print-4 .code:n     = \keys_precompile:neN { enumext / level-4 }
4486                             { \__enumext_filter_save_key:n {#1} }
4487                             \l__enumext_print_keyans_iv_tl,
4488     print-4 .initial:n  = { nosep, label=\Alph*., first=\small, font=\small },
4489     print-* .code:n     = \keys_precompile:neN { enumext / enumext* }
4490                             { \__enumext_filter_save_key:n {#1} }
4491                             \l__enumext_print_keyans_vii_tl, % starred nested
4492     print-* .initial:n  = { nosep, label=\arabic*., first=\small, font=\small },
4493   }
```

💣 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.

```
4494 \NewDocumentCommand \printkeyans { s O{} m }
4495   {
4496     \group_begin:
4497       \tl_use:N \l__enumext_print_keyans_i_tl
4498       \tl_use:N \l__enumext_print_keyans_ii_tl
4499       \tl_use:N \l__enumext_print_keyans_iii_tl
4500       \tl_use:N \l__enumext_print_keyans_iv_tl
4501       \tl_use:N \l__enumext_print_keyans_vii_tl
4502       \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
4503     \group_end:
4504   }
```

(*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.

```
4505 \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
4506   {
4507     \seq_if_exist:cTF { g__enumext_#3_seq }
4508       {
4509         \seq_if_empty:cF { g__enumext_#3_seq }
4510           {
4511             %%\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*⟩.

```
4512          \bool_if:nTF {#1}
4513            {
4514              \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
4515                {
4516                  \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
4517                }
4518                {
4519                  \tl_use:N \l__enumext_print_keyans_starred_tl
4520                  \begin{enumext*}[#2]
4521                    \keys_set:nn { enumext / level-1 }{ base-fix }
4522                    \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4523                  \end{enumext*}
4524                }
4525            }
```

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*⟩.

```
4526            {
4527              \begin{enumext}[#2]
4528                \keys_set:nn { enumext / enumext* }{ base-fix }
4529                \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
4530              \end{enumext}
4531            }
4532          }
4533        }
4534        {
4535          \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
4536        }
4537    }
```

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

11.46 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.

```
4538 \cs_new:Npn \__enumext_filter_first_level:n #1
4539   {
4540     \use:e
4541       {
4542         \keyval_parse:NNn
4543           \__enumext_filter_first_level_key:n
4544           \__enumext_filter_first_level_pair:nn {#1}
4545       }
4546   }
```

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*.

```
4547 \cs_new:Npn \__enumext_filter_first_level_key:n #1
4548   {
4549     \str_case:nnF {#1}
4550       {
4551         { resume    } {}
4552         { resume*   } {}
4553       }
4554       { , { \exp_not:n {#1} } }
4555   }
```

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.

```
4556 \cs_new:Npn \__enumext_filter_first_level_pair:nn #1#2
4557   {
4558     \str_case:nnF {#1}
4559       {
```

```
4560            { series } {}
4561            { resume } {}
4562            { save-ans } {}
4563          }
4564        { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
4565    }
```

(*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`.

```
4566  \keys_define:nn { enumext / meta-families }
4567    {
4568      enumext-1 .code:n =
4569                {
4570                  \keys_set:ne { enumext / level-1  }
4571                    {
4572                      \__enumext_filter_first_level:n {#1}
4573                    }
4574                } ,
4575      enumext-2 .code:n = { \keys_set:nn { enumext / level-2  } {#1} } ,
4576      enumext-3 .code:n = { \keys_set:nn { enumext / level-3  } {#1} } ,
4577      enumext-4 .code:n = { \keys_set:nn { enumext / level-4  } {#1} } ,
4578      keyans    .code:n = { \keys_set:nn { enumext / keyans   } {#1} } ,
4579      enumext*  .code:n =
4580                {
4581                  \keys_set:ne { enumext / enumext* }
4582                    {
4583                      \__enumext_filter_first_level:n {#1}
4584                    }
4585                } ,
4586      keyans*   .code:n = { \keys_set:nn { enumext / keyans*  } {#1} } ,
4587      print*    .code:n = { \keys_set:nn { enumext / print    } { print*  = {#1} } } ,
4588      print-1   .code:n = { \keys_set:nn { enumext / print    } { print-1 = {#1} } } ,
4589      print-2   .code:n = { \keys_set:nn { enumext / print    } { print-2 = {#1} } } ,
4590      print-3   .code:n = { \keys_set:nn { enumext / print    } { print-3 = {#1} } } ,
4591      print-4   .code:n = { \keys_set:nn { enumext / print    } { print-4 = {#1} } } ,
4592      print-*   .code:n = { \keys_set:nn { enumext / print    } { print-* = {#1} } } ,
4593      unknown   .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
4594    }
```

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

```
4595  \seq_const_from_clist:Nn \c__enumext_all_families_seq
4596    {
4597      enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
4598      keyans*, print-1, print-2, print-3, print-4, print-*, print*,
4599    }
```

`\setenumext` Now we define the user command `\setenumext`.

```
4600  \NewDocumentCommand \setenumext { O{enumext,1} +m }
4601    {
4602      \tl_if_novalue:nTF {#1}
4603        {
4604          \seq_map_inline:Nn \c__enumext_all_families_seq
4605        }
4606        {
4607          \seq_clear:N \l__enumext_setkey_tmpa_seq
4608          \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
4609          \int_set:Nn \l__enumext_setkey_tmpa_int
4610            {
4611              \seq_count:N \l__enumext_setkey_tmpb_seq
4612            }
4613          \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
4614            {
4615              \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
4616              \seq_map_function:NN \l__enumext_setkey_tmpb_seq \__enumext_set_parse:n
4617              \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
4618                {
4619                  \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
4620                }
4621            }
4622            {
```

```
4623            \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
4624          }
4625        \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
4626          { \seq_map_inline:Nn \c__enumext_all_families_seq }
4627          { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
4628      }
4629      {
4630        \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
4631      }
4632  }
```

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

__enumext_set_parse:n
__enumext_set_error:nn

Internal functions used by the \setenumext command.

```
4633  \cs_new_protected:Npn \__enumext_set_parse:n #1
4634    {
4635      \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
4636      \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
4637        { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
4638      \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
4639        {
4640          \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
4641            { \tl_trim_spaces:n {#1} }
4642        }
4643        { \__enumext_set_error:nn {#1} { } }
4644    }
4645  \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
4646    { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }
```

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

11.47 Messages

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

```
4647  \msg_new:nnn { enumext } { package-load }
4648    {
4649      The ~ '#1' ~ package ~ is ~ already ~ loaded.
4650    }
4651  \msg_new:nnn { enumext } { package-not-load }
4652    {
4653      The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
4654    }
4655  \msg_new:nnn { enumext } { package-load-foot }
4656    {
4657      The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
4658    }
```

Message used in the creation of counters by enumext package.

```
4659  \msg_new:nnn { enumext } { counters }
4660    {
4661      The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
4662      package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
4663    }
```

Message used by align and mark-pos keys.

```
4664  \msg_new:nnn { enumext } { unknown-choice }
4665    {
4666      The ~ value ~ '#3' ~ for ~ '#1' ~ key ~ is ~ invalid ~ use ~ ('#2').
4667    }
```

Message used by reserved anskey* environment by enumext package.

```
4668  \msg_new:nnnn { enumext } { anskey-env-error }
4669    {
4670      The ~ '#1' ~ environment ~is ~ reserved ~ by ~\\
4671      'enumext' ~ package, ~ It~ is~ already~ defined.
4672    }
4673    {
4674      The ~ anskey* ~ environment ~ is ~ defined ~ internally ~
4675      for ~ the ~ 'save-ans' ~ key.\\
4676    }
```

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

```
4677 \msg_new:nnn { enumext } { store-prop }
4678   {
4679     * ~ Package ~ enumext: ~ Creating ~
4680     \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
4681   }
4682 \msg_new:nnn { enumext } { store-seq }
4683   {
4684     * ~ Package ~ enumext: ~ Creating ~
4685     \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
4686   }
4687 \msg_new:nnn { enumext } { store-int }
4688   {
4689     * ~ Package ~ enumext: ~ Creating ~
4690     \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_context:.
4691   }
4692 \msg_new:nnn { enumext } { prop-seq-int-hook }
4693   {
4694     * ~ Package ~ enumext: ~ Elements ~ in ~
4695     \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
4696     * ~ Package ~ enumext: ~ Elements ~ in ~
4697     \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
4698     * ~ Package ~ enumext: ~ Value ~ off ~
4699     \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
4700   }
4701 \msg_new:nnn { enumext } { item-answer-hook }
4702   {
4703     * ~ Package ~ enumext: ~ Value ~ off ~
4704     \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\\
4705     * ~ Package ~ enumext: ~ Value ~ off ~
4706     \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\\
4707     * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
4708   }
```

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

```
4709 \msg_new:nnn { enumext } { invalid-key }
4710   {
4711     The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
4712   }
4713 \msg_new:nnn { enumext } { unknown-key-family }
4714   {
4715     Unknown~key~family~`\l_keys_key_str'~for~enumext.
4716   }
```

Messages used in length calculation.

```
4717 \msg_new:nnn { enumext } { width-negative }
4718   {
4719     Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\\
4720     The ~ key ~ '#1'~ accepts ~ values  ~ >= ~ 0pt.
4721   }
4722 \msg_new:nnn { enumext } { width-zero }
4723   {
4724     Invalid ~ '#1=#2' ~ \msg_line_context:.\\
4725     The ~ key ~ '#1'~ accepts ~ values  ~ > ~ 0pt.
4726   }
```

Messages used by show-length key in enumext.

```
4727 \msg_new:nnn { enumext } { list-lengths }
4728   {
4729     **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
4730     \__enumext_show_length:nnn { dim  } { labelsep      } {#1}
4731     \__enumext_show_length:nnn { dim  } { labelwidth    } {#1}
4732     \__enumext_show_length:nnn { dim  } { itemindent    } {#1}
4733     \__enumext_show_length:nnn { dim  } { leftmargin    } {#1}
4734     \__enumext_show_length:nnn { dim  } { rightmargin   } {#1}
4735     \__enumext_show_length:nnn { dim  } { listparindent } {#1}
4736     \__enumext_show_length:nnn { skip } { topsep     } {#1}
4737     \__enumext_show_length:nnn { skip } { parsep     } {#1}
4738     \__enumext_show_length:nnn { skip } { partopsep } {#1}
4739     \__enumext_show_length:nnn { skip } { itemsep    } {#1}
4740     **********************************************
4741   }
```

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

```
4742  \msg_new:nnn { enumext } { list-lengths-not-nested }
4743    {
4744      **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
4745      \__enumext_show_length:nnn { dim  } { labelsep    } {#1}
4746      \__enumext_show_length:nnn { dim  } { labelwidth  } {#1}
4747      \__enumext_show_length:nnn { dim  } { itemindent  } {#1}
4748      \__enumext_show_length:nnn { dim  } { leftmargin  } {#1}
4749      \__enumext_show_length:nnn { dim  } { rightmargin } {#1}
4750      \__enumext_show_length:nnn { dim  } { listparindent } {#1}
4751      \__enumext_show_length:nnn { skip } { topsep    } {#1}
4752      \__enumext_show_length:nnn { skip } { parsep    } {#1}
4753      \__enumext_show_length:nnn { skip } { partopsep } {#1}
4754      \__enumext_show_length:nnn { skip } { itemsep   } {#1}
4755      *****************************************************
4756    }
```

Messages used by `ref` key.

```
4757  \msg_new:nnn { enumext } { key-ref-empty }
4758    {
4759      Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4760    }
```

Messages used by `save-ans` key.

```
4761  \msg_new:nnn { enumext } { save-ans-empty }
4762    {
4763      Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
4764    }
4765  \msg_new:nnn { enumext } { save-ans-log }
4766    {
4767      * ~ Package ~ enumext: ~ Start ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
4768    }
4769  \msg_new:nnn { enumext } { save-ans-log-hook }
4770    {
4771      * ~ Package ~ enumext: ~ Stop ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
4772    }
4773  \msg_new:nnn { enumext } { save-ans-hook }
4774    {
4775      Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
4776    }
```

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

```
4777  \msg_new:nnn { enumext } { need-save-ans }
4778    {
4779      Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
4780    }
4781  \msg_new:nnn { enumext } { items-same-answer }
4782    {
4783      ******************************************\\
4784      * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~
4785      for ~ \c_left_brace_str #2 \c_right_brace_str\\
4786      * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~
4787      'OK', ~ all ~ items ~ with ~ answer.\\
4788      ******************************************
4789    }
4790  \msg_new:nnn { enumext } { item-greater-answer }
4791    {
4792      Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
4793      started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
4794      Items ~ > ~ Answers.
4795    }
4796  \msg_new:nnn { enumext } { item-less-answer }
4797    {
4798      Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
4799      started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
4800      Items ~ < ~ Answers.
4801    }
```

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

```
4802  \msg_new:nnn { enumext } { missing-starred }
4803    {
4804      Missing ~ '\c_backslash_str #1*' ~ #2.
4805    }
```

```
4806  \msg_new:nnn { enumext } { many-starred }
4807    {
4808      Many ~ '\c_backslash_str #1*' ~ #2.
4809    }
```

Messages used by \printkeyans* command.

```
4810  \msg_new:nnn { enumext } { print-starred }
4811    {
4812      \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
4813      #2 ~ environment ~  \msg_line_context:.
4814    }
```

Message for the nesting depth of the environment enumext.

```
4815  \msg_new:nnn { enumext } { list-too-deep }
4816    {
4817      Too ~ deep ~ nesting  ~ for ~ 'enumext' ~ \msg_line_context:.~ \\
4818      The ~ maximum  ~ level  ~ of  ~ nesting  ~ is ~ 4.
4819    }
```

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

```
4820  \msg_new:nnn { enumext } { anskey-unnumber-item }
4821    {
4822      Can't ~ store ~ with ~ a ~ unnumbered ~ \c_backslash_str item ~ \msg_line_context:.
4823    }
4824  \msg_new:nnn { enumext } { anskey-already-stored }
4825    {
4826      Content ~ already ~ stored ~ for ~ this ~ \c_backslash_str item ~ \msg_line_context:.
4827    }
4828  \msg_new:nnn { enumext } { anskey-empty-arg }
4829    {
4830      Can't ~ store ~ empty ~ content ~ ~ \msg_line_context:.
4831    }
4832  \msg_new:nnn { enumext } { anskey-wrong-place }
4833    {
4834      Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
4835      '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4836    }
4837  \msg_new:nnn { enumext } { anskey-nested }
4838    {
4839      The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
4840    }
4841  \msg_new:nnn { enumext } { anskey-math-mode }
4842    {
4843      #1 ~ can't ~ work ~ in ~ math ~ mode ~ \msg_line_context:.
4844    }
4845  \msg_new:nnn { enumext } { anskey-env-wrong }
4846    {
4847      The ~ environment ~ anskey* ~ cannot ~ use ~ in ~ '#1' ~ \msg_line_context:.
4848    }
4849  \msg_new:nnn { enumext } { anspic-wrong-place }
4850    {
4851      Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
4852      '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
4853    }
4854  \msg_new:nnn { enumext } { command-wrong-place }
4855    {
4856      Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
4857      '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
4858    }
4859  \msg_new:nnnn { enumext } { anskey-env-key-unknown }
4860    {
4861      The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
4862      'anskey*' ~ and ~ is ~ being ~ ignored.
4863    }
4864    {
4865      The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
4866      Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
4867    }
4868  \msg_new:nnnn { enumext } { anskey-env-key-value-unknown }
4869    {
4870      The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
4871      'anskey*' ~ and ~ is ~ being ~ ignored.
```

```
4872        }
4873        {
4874          The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
4875          Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
4876        }
4877 \msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
4878      { The~key~'#1'~is~unknown~by~'\c_backslash_str anskey'~and~is~being~ignored.}
4879      {
4880        The~command~'\c_backslash_str anskey'~does~not~have~a~key~called~'#1'.\\
4881        Check~that~you~have~spelled~the~key~name~correctly.
4882      }
4883 \msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
4884      { The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ igno
4885      {
4886        The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
4887        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
4888      }
```

Messages used by keyans and keyanspic environment.

```
4889 \msg_new:nnn { enumext } { keyans-nested }
4890      {
4891        The ~ environment ~ 'keyans' ~ can't ~ be  ~ nested  ~ \msg_line_context:.
4892      }
4893 \msg_new:nnn { enumext } { keyans-wrong-level }
4894      {
4895        Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:.~ \\
4896        The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4897      }
4898 \msg_new:nnn { enumext } { wrong-place }
4899      {
4900        Wrong ~ place ~ for ~ '#1' ~ environment ~\msg_line_context:.~ \\
4901        '#1' ~ is ~ only ~ found ~ with ~ '#2' ~  in  ~  'enumext.
4902      }
4903 \msg_new:nnn { enumext } { keyanspic-nested }
4904      {
4905        The ~ environment ~ 'keyanspic' ~ can't ~ be  ~ nested~ \msg_line_context:.~.
4906      }
4907 \msg_new:nnn { enumext } { keyanspic-wrong-level }
4908      {
4909        Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:.~ \\
4910        The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
4911      }
4912 \msg_new:nnn { enumext } { keyanspic-item-cmd }
4913      {
4914        Can't ~ use  ~ \c_backslash_str item ~ in ~ keyanspic ~ \msg_line_context:.
4915      }
```

Messages used by \getkeyans command.

```
4916 \msg_new:nnn { enumext } { undefined-storage-anskey }
4917      {
4918        Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
4919      }
```

Messages used by \miniright command.

```
4920 \msg_new:nnn { enumext } { missing-miniright }
4921      {
4922        Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\\
4923        The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
4924      }
4925 \msg_new:nnn { enumext } { wrong-miniright-place }
4926      {
4927        Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
4928        Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
4929      }
4930 \msg_new:nnn { enumext } { wrong-miniright-use }
4931      {
4932        Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
4933        '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
4934      }
```

Messages used by enumext* and keyans* environments.

```
4935 \msg_new:nnn { enumext } { nested }
4936      {
```

```
4937        The ~ starred ~ environment ~ can't ~ be ~ nested ~ \msg_line_context:.
4938      }
4939  \msg_new:nnn { enumext } { item-joined }
4940    {
4941       Items ~ joined ~ (#1) ~ > ~ #2  ~ columns ~\msg_line_context:.
4942    }
4943  \msg_new:nnn { enumext } { item-joined-columns }
4944    {
4945       Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.
4946    }
```

11.48   Finish package

Finish package implementation.

```
4947  \file_input_stop:
4948  ⟨/package⟩
```

12 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.