enumext
ENUMERATE EXERCISE SHEETS

V1.2   2025-03-28[*]

©2024–2025 by Pablo González[†]

Abstract

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

Contents

Motivation and acknowledgments

Usually it is enough to use the classic enumerate environment to generate *"simple exercise sheets"* or *"multiple choice questions"*, the basic idea behind enumext is to cover three points:

1. To have a simple interface to be able to write *"lists of exercises"* with *"answers"*.
2. To have a simple interface for writing *"multiple choice questions"*.
3. To have a simple interface for placing *"columns"* and *"drawings"* or *"tables"*.

This package would not be possible without Phelype Oleinik who has collaborated and adapted a large part of the code and all LaTeX team for their great work and to the different members of the TeX-SX community who have provided great answers and ideas. Here a note of the main ones:

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

---

License and Requirements

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

The `enumext` package loads and requires `multicol`[3] and `scontents`[4] packages, need to have a modern TeX distribution such as TeX Live or MiKTeX. It has been tested with the standard classes provided by LaTeX: `book`, `report`, `article` and `letter` on `10pt`, `11pt` and `12pt`.

💣 The minimum requirement is LaTeX release 2024-11-01.

1 Introduction

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

In my work (and in the work of many teachers) it is common to use *"simple exercise sheets"* also known as *"informal lists of exercises"*, 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)
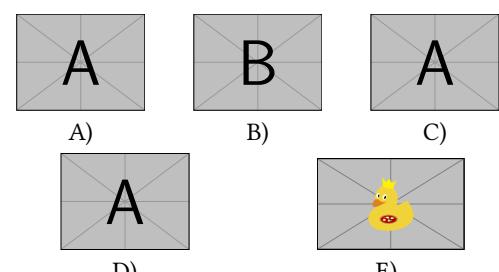
5. Question with image on right side:

A) value
B) value
C) value
D) correct
E) value
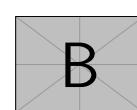
Where what we are interested in the ⟨*label*⟩ and a *"short note"* that we leave as an explanation, and then print them:

1. B), $x = 5$
2. D)
3. C), some note

4. E), A duck
5. D), "other note"

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

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

1.1 Description and usage

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

This package is NOT 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 luatex enumext.ins and move all files to appropriate locations, then run mktexlsr. To produce the documentation run arara enumext.dtx.

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

The package is loaded in the usual way:

```
\usepackage{enumext}
```

1.2 The concept of left margin

There is a direct relationship between the parameters \leftmargin, \itemindent, \labelwidth and \labelsep plus an *"extra space"* 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.
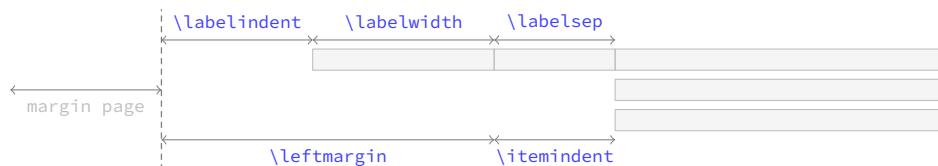
Figure 1: Representation of horizontal lengths in enumitem.

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.
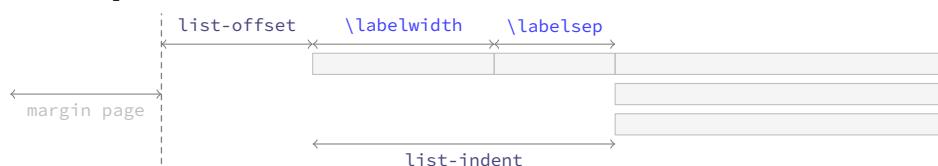
Figure 2: Representation of horizontal lengths concept in enumext.

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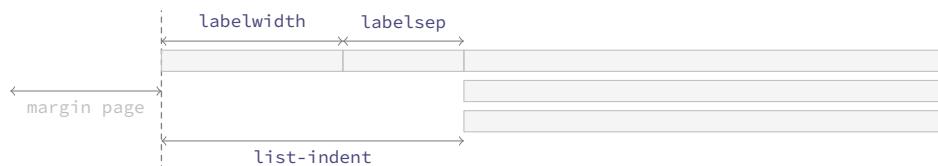
Figure 3: Default horizontal lengths list-offset=0pt, list-indent=\labelwidth+\labelsep in enumext.

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 and \foreachkeyans to print all *stored content*, \miniright for minipage, \setenumext and \setenumextmeta to config [⟨*key = val*⟩] options.

1.3.1 Internal counters

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

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

1.3.2 Public dimension

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

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

1.3.3 Support for multicol

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

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

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

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

1.3.4 Support for minipage

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

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

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

1.3.5 The \label and \ref system

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

1.3.6 Support for \footnote

The enumext* and keyans* environments and the mini-env key use the minipage environment in their implementation but in a transparent way for the user, i.e. it is only used for typesetting and not directly. The enumext package provides an *internal implementation* for the command \footnote compatible with the hyperref package to work in the same way as if it were used anywhere in the document.

Unfortunately, if *tagging* PDF is not enabled, it will not produce the expected *"links"* because the internal implementation uses `\footnotetext[⟨number⟩]` and `\footnotemark[⟨number⟩]{⟨text⟩}` and support for these is limited by the `hyperref` package.

The best way to solve this if *tagged* PDF is NOT active 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). Load it is as follows:

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

🏷 At the moment the `footnotehyper` package is not compatible with *tagged* PDF.

2   The environments provided

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

<table>
<tr><td>enumext
enumext*</td><td>

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

</td><td>

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

</td></tr>
</table>

2.1   The environment enumext

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

Example with columns=2

1. This text is in the first level.

  (a) This text is in the second level.

     i. This text is in the third level.

A. This text is in the fourth level.

X This text is in the first level.

⋆ 2. This text is in the first level.

2.2   The environment enumext*

The enumext* is a *horizontal list environment* similar to the shortenumerate or tasks environments provided by the shortlst[15] and tasks[16] packages, `\item` and `\item[⟨custom⟩]` work as usual. The options can be configured globally using `\setenumext` command and locally using `[⟨key = val⟩]` in the environment.

Some considerations to take into account for this environment:

- The environment cannot be nested within itself or in the environment keyans*, but it can be nested within enumext and vice versa.
- Each *"item content"* in the environment is placed within a minipage environment whose *width* is stored in the dimension `\itemwidth` that NOT includes labelwith, labelsep, only the *width of the content*.
- You cannot have floating environments like figure or table but `\footnote` with hyperref support is supported if the footnotehyper package is loaded (see §1.3.6 for full support).
- You cannot have any standard list environments like itemize, enumerate, description, quote, quotation, verse, center, flushleft, flushright, verbatim, tabbing, trivlist, list and all environments created with `\newtheorem`.

Example with columns=2

1. This text is in the first level.

X This text is in the first level.

2. This text is in the first level.

⋆ 4. This text is in the first level.

2.3   The command `\item*`

<table>
<tr><td>\item*</td><td>`\item*[⟨symbol⟩][⟨offset⟩]`</td></tr>
</table>

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

The *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: \textborn

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*={$\star$}.

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

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

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

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

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

7. The seventh X The eighth item is way too long for this and needs two columns (196.17749pt) 9. The ninth

Z The tenth (89.28171pt)

3 The command \setenumext

\setenumext

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

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

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

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

4 The command \setenumextmeta

\setenumextmeta

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

The command \setenumextmeta adds a new *"meta-key"* for the environments enumext and enumext*, the {⟨*key name*⟩} must be different from those defined by the package. If the *optional argument* is not passed, the new *"meta-key"* will be created for the *"first level"* of the environment enumext.

The *starred argument* '*' will create the new *"meta-key"* for the environment enumext* and for all levels of the environment enumext. For example: \setenumextmeta*{midsep}{topsep=3pt, partopsep=0pt} will create a new key midsep available for all levels of the enumext environment and the enumext* environment and we can use it like any other key so \begin{enumext}[midsep] and \begin{enumext*}[midsep] will be valid.

5 The `keyval` system

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

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

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

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

5.1 Keys for `label` and `ref`

mode-box ⟨*value forbidden*⟩   default: *not used*

This is a *"switch-key"* that does not receive an argument and is *"only"* available for the *"first level"* of the enumext environment and the enumext* environment. When this is set the `label`, `font`, `wrap-label` and `wrap-label*` keys are executed within `\makebox` for the enumext and keyans environments.

🛈 This key is intended for compatibility with *tagged* PDF and is forcibly *"enabled"* when `\DocumentMetadata` is present. If you want to get the same document output whether `\DocumentMetadata` is active or not, you must enable this key.

🛈 In the enumext* and keyans* environments `\makelabel` are redefined using `\makebox` by default. If enumext or keyans is used in the enumext* environment the key must be activated manually.

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

Sets the ⟨*label*⟩ that will be printed at the *current level* and default value for `labelwidth` key. The default value for the first level of the environments enumext and enumext* 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 label and ref"* system with the `save-ref` key. You cannot use commands with ⟨*label*⟩ as an argument, for example `\emph{\alph*}` will return an error. For full customization of how ⟨*label*⟩ is displayed use the `font`, `wrap-label` and/or `wrap-label*` keys.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

5.2 Keys for spaces

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

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

5.2.1 Vertical spaces

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

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

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

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

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

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

Set the *vertical space* added, beyond `topsep`, to the "top" and "bottom" of the entire environment if the environment instance is preceded by a *"blank line"* or `\par` command. Internally sets the value of `\partopsep` for the current level. The default values for first and second level in environment `enumext` are `2.0pt plus 1.0pt minus 1.0pt`, for third and fourth level are `1.0pt minus 1.0pt`. For the `keyans` environment the default value is `2.0pt plus 1.0pt minus 1.0pt`, and for the `keyans*` and `enumext*` environments it is available but *without* effect.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

5.2.2 Horizontal spaces

`list-offset` = {⟨*rigid length*⟩}   default: *0pt*

Sets the *horizontal translation* of the entire environment level from the left edge of the box defined by the `labelwidth` key. Internally sets the values of `\leftmargin` and `\itemindent` for the current level.

`list-indent` = {⟨*rigid length*⟩}   default: *labelwidth + labelsep*

Sets the *indentation* of the whole environment under the box defined by `labelwidth` and `labelsep` keys. Internally sets the value of `\leftmargin` and `\itemindent` for the current level. If `list-indent=0pt` is set in the environments `enumext` and `keyans` the ⟨*label*⟩ will be part of the text, separated by the value of the `labelsep` key and the *first word*, in simple terms it will look like a *"common paragraph"*.

The `enumext*` and `keyans*` environments are implemented using `\makebox` and `minipage` which causes *"list indent"* to always be equal to the value passed to `labewdith` plus `labelsep`. Passing a value to this key is equivalent to setting the value for the `list-offset` key.

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

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

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

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

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

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

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

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

5.3 Keys for add code

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

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

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

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

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

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

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

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

🎯 In the enumext* and keyans* environments this key is executed after the listparindent, parsep and itemindent keys within the minipage environment in which the *"item content"* is placed.

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

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

5.4 Keys for start, series and resume

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

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

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

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

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

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

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

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

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

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

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

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

5.5 Keys for multicols

columns = {⟨*integer*⟩} default: *1*

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

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

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

5.6 Keys for `minipage`

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

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

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

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

5.6.1 The command `\miniright`

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

The `\miniright` command close the `minipage` environment on the *"left side"* and opens the `minipage` environment on the *"right side"* by starting it with the `\centering` command. It must be placed *"after"* the last `\item` of the current environment and *"before"* starting the material to be placed on the *"right side"*.

The *starred argument* '`*`' inhibits the use of `\centering` command i.e. the usual LATEX justification is maintained in the `minipage` on the *"right side"*.

5.6.2 The key `mini-right`

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

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

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

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

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

6 The storage system

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

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

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

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

For security reasons the *optional argument* of the inner levels or the nested environment are *filtered* by excluding all ⟨*keys*⟩ related to the *"storage system"* (§6.1) along with the keys `mini-env`, `mini-sep`, `mini-right`, `mini-right*`, `series`, `resume` and `resume*` when storing in *sequence* {⟨*store name*⟩} set by `save-ans` key.

6.1 Keys for storage system

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

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

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

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

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

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

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

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

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

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

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

6.1.1   Keys for `label` and `ref`

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

Activates the *"internal label and ref"* mechanism for referencing *"stored content"* in *prop list* {⟨*store name*⟩} set by `save-ans` key. To reference the location of the *"stored content"* within the environment you must use `\ref`{⟨*store name : position*⟩}, where ⟨*position*⟩ corresponds to the position occupied by the *"stored content"* in the *prop list* {⟨*store name*⟩} returned by the `show-pos` key. For example `\ref`{`test:4`} will return `3.(b)` which corresponds to the location of the *"stored content"* at position 4 in *prop list* `test` within the environment in which the key `save-ans=test` was set.

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

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

6.1.2   Keys for `wrap` and `marks`

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

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

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

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

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

Keys for `\anskey` and `anskey*`

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

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

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

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

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

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

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

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

Keys for keyans, keyans* and keyanspic

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

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

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

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

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

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

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

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

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

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

6.1.3 Keys for debug and checking

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

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

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

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

6.2 The command \anskey

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

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

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

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

6.2.1 Keys for \anskey

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

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

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

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

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

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

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

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

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

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

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

Example

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

★ 1. Text containing our instructions or questions.
    ∗ | first answer |
2. Text containing our instructions or questions.
    (a) Question.
      ∗ | second answer |

3. Text containing our instructions or questions.
    ∗ | third answer |
4. Text containing our instructions or questions.
    ∗ | fourth answer |

6.3 The environment anskey*

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

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

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

The anskey* environment is implemented using the scontents package, \begin{anskey*} and \end{anskey*} must be in different lines, all ⟨*keys*⟩ must be passed separated by commas and "without separation" of the start of the environment.

Comments "%" or "any character" after \begin{anskey*} or [⟨*key = val*⟩] on the same line are NOT supported, the package scontents will return an "error" message if this happens. In a similar way comments "%" or "any character" after \end{anskey*} on the same line the package scontents will return a "warning" message.

6.3.1 Keys for anskey*

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

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

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

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

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

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

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

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

Example

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

    \begin{anskey*}[item-star]
      ⟨first answer⟩
    \end{anskey*}

  \item Text containing our instructions or questions.

    \begin{enumext}
      \item Question.
        \begin{anskey*}
          ⟨second answer⟩
        \end{anskey*}
    \end{enumext}

  \item Text containing our instructions or questions.

    \begin{anskey*}
      ⟨third answer⟩
    \end{anskey*}

  \item Text containing our instructions or questions.

    \begin{anskey*}
      ⟨fourth answer⟩
    \end{anskey*}
\end{enumext}
```

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

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

6.4 The environments keyans and keyans*

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

The `keyans` and `keyans*` environments are *"enumerated list"* environments designed for *"multiple choice"* questions activated by the `save-ans` key. This environments can NOT be nested and must always be at the *"first level"* of the `enumext` environment, the commands `\item` and `\item[⟨custom⟩]` work in the usual and the command `\item(⟨columns⟩)` is available for the `keyans*` environment.

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

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

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

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

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

6.4.1 The `\item*` in keyans and keyans*

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

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

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

Example

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

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

  \item Text containing a question and image.

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

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

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

Some text

6.5 The environment keyanspic

`keyanspic`
```
\begin{keyanspic}[⟨key = val⟩] \anspic*[⟨content⟩]{⟨drawing or tabular⟩} \end{keyanspic}
```

The `keyanspic` environment is an *"enumerated list"* environment activated by the `save-ans` key that has the same configuration for *"spacing"* and ⟨*label*⟩ as the `keyans` environment that uses the `\anspic` command instead of `\item`. It is intended for placing *drawings or tabular* with ⟨*label*⟩ centered *above* or *below* in a *single line* or *upper and lower* layout style.

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

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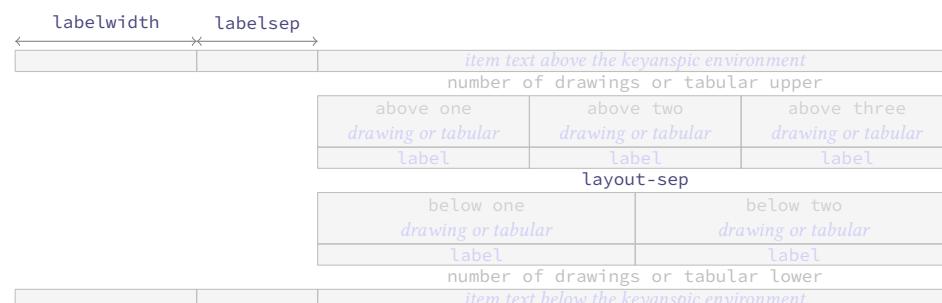
Figure 6: Representation of the `keyanspic` environment with `layout-sty=`{⟨*3, 2*⟩} in `enumext`.

This environment cannot be nested and must *always* be at the *"first level"* of the enumext environment, the \item command is disabled and ⟨*keys*⟩ cannot be set using \setenumext.

6.5.1 Keys for keyanspic

label-pos = {⟨*above* | *below*⟩}   default: *below*

Set the *position* of ⟨*label*⟩ to be centered "above" or "below" *drawings* or *tabular* when the \anspic command is executed.

label-sep = {⟨*rubber length* | *rigid length*⟩}   default: *internal adjustment*

Set the *vertical spacing* between the ⟨*label*⟩ centered "above" or "below" and *drawings* or *tabular* when running the \anspic command.

layout-sty = {⟨*n° upper , n° lower*⟩}   default: *not set*

Set the *number* of *drawings* or *tabular* that will be distributed "upper" and "lower" within the environment when executing the \anspic command. The value must be passed in braces and if not set or the ⟨*n° lower*⟩ is omitted the *drawings* or *tabular* will be put on a *single line.*

layout-sep = {⟨*rubber length* | *rigid length*⟩}   default: *adjusted parsep from keyans*

Set the *vertical separation* between the number of *drawings* or *tabular* placed at the "upper" and "lower" within the environment when executing the \anspic command. Internally adjusts the parsep value taken from the keyans environment.

layout-top = {⟨*rubber length* | *rigid length*⟩}   default: *adjusted topsep from keyans*

Set the *vertical space* added to both the top and bottom of the environment. Internally adjust the value of topsep taken from keyans environment.

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

6.5.2 The command \anspic

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

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

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

Example

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

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

  \item Question with images and labels above.

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

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

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

\end{enumext}
```

1. Question with images and labels below.


A)

B)

C)

D)

∗ E) [note]
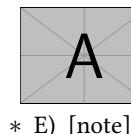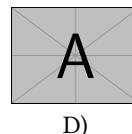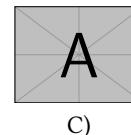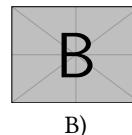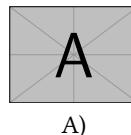
2. Question with images and labels above.

A)

B)

C)

D)

∗ E) [note]
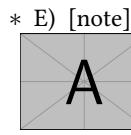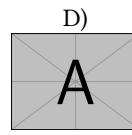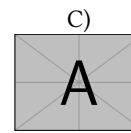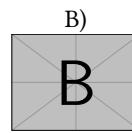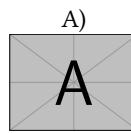
3. Question with images and labels below on a single line.

A)

B)

C)

D)

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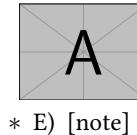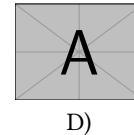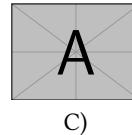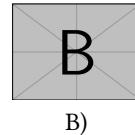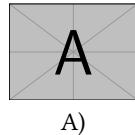
🏷 Remember to pass the `alt={⟨description⟩}` key to the `\includegraphics` command when creating a *tagged* PDF.

6.6   Printing stored content

6.6.1   The command `\getkeyans`

`\getkeyans`

`\getkeyans{⟨store name : position⟩}`

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

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

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

6.6.2   The command `\foreachkeyans`

`\foreachkeyans`

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

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

Options for command

`sep = {⟨code⟩}` default: *{; }*

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

`step = {⟨integer⟩}` default: *1*

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

`start = {⟨integer⟩}` default: *1*

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

`stop = {⟨integer⟩}` default: *0*

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

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

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

after = {⟨*code*⟩}   default: *empty*

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

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

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

6.6.3 The command `\printkeyans`

`\printkeyans`

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

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

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

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

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

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

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

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

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

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

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

Example

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

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

  \item Related to Linux

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

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

\printkeyans{sample}
```

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

[1] $\boxed{3(x + y + z)}$

2. True False

   (a) LaTeX2e is cool?

   [2] $\boxed{\text{Very True!}}$

3. Related to Linux

   (a) You use linux?

   [3] $\boxed{\text{Yes}}$

   (b) Rate the following package and class

      i. `xsim`

      [4] $\boxed{\text{very good}}$

      ii. `exsheets`

      [5] $\boxed{\text{obsolete}}$

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

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

7 Full examples

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

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

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

Example 1

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

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

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

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

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

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

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

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

Example 2

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

---

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

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

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

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

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

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

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

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

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

1. B   ※ 2. A   ※
3. B   ※ 4. A   ※

Example 3

A *"simple multiple choice"* test 📄.

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

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

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

4. Question with image and label below:

   Ⓐ   Ⓑ   Ⓒ   Ⓓ   Ⓔ

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

Test keys

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

Example 4

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

🦆 Factor $x^2 - 2x + 1$

🦆 Factor $3x + 3y + 3z$

The following questions need to be cuaqtified :)

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

🦆 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.

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.

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.

4

Which choice best describes what happens in the passage?

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

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

Example 6

Adapted from the response to Environment for enumerate environment 📄.

8.5a, KSC **10.** sample

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

9.5a, KSC **11.** sample

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

12. sample

    A sample
    B answer
    C sample
    D sample

13. sample

    A sample
    B sample
    C sample
    D answer

10. B(8.5a, KSC)
11. D(9.5a, KSC)
12. B(10.5a, KSC)
13. D(11.5a, KSC)

8 Tagged PDF examples

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

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

```
\DocumentMetadata
  {
    lang = en-US, pdfversion = 2.0, pdfstandard = ua-2,
    testphase = {phase-III, math, title, table, firstaid},
  }
```

All examples have been checked using veraPDF together with ngpdf.

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

9 The way of non-enumerated lists

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

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

Fake itemize environment

Here we set the label key using the default settings in LaTeX for the four levels \textbullet, \textendash, \textasteriskcentered and \textperiodcentered together with the nosep key to reduce the vertical spaces in the left side example and set the label key in *mathematical mode* for the right side as \ast, \diamond, \circ and \star for the four levels together with the nosep key

- First level item
  - Second level item
    * Third level item
      · Fourth level item
- First level item

∗ First level item
  ⋄ Second level item
    ∘ Third level item
      ⋆ Fourth level item
∗ First level item

Fake description environment

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

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

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

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

- The small space at the beginning of the *"unlabeled entry"* corresponds to `\labelsep` and can be removed using `\hspace{-\labelsep}` at the beginning of the line.
- When *tagged* PDF is active the default `description` style is NOT available due to the redefinition of `\makelabel` for the `align` key which uses `\makebox` in this case, meaning that `\item[⟨content⟩]` will not extend beyond `\labelwidth` which causes overlaps,

Description indented by label

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

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

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

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

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

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

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

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

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

Description with multi-line labels

The `label` key does not accept *multiline material*, this is where the `wrap-label` and `wrap-label*` keys comes into play. Unlike the `enumitem` package, the `align` key only supports three options, so what we will do is create a command in the style `\parleft` of `enumitem` that allows us to place *multiline labels* using `\parbox`.

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

Now we just need to set `wrap-label*={\labelbx{#1}}`.

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

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

Final notes

The original implementation (if you can call it that) of the ideas that led to the creation of enumext were some macros using the enumerate[5] package for personal use created in early 2003, the code was quite questionable, but functional for these simple requirements.

With the great answers given by Christian Hupfer in Create a fake label ref using list and the answer given by David Carlisle in Change the use of label ref by data save in an array (list) I managed to create a more solid code than the original version, now using the l3prop[11] and l3seq[11] modules together with the hyperref[8] and enumitem[6] packages, which did the job, but with some limitations.

As time went by I took these limitations as a personal challenge which I called *"reinventing the wheel"*, since there were packages and classes that did more or less what I was looking for, but did not fit my simple requirements. This *"reinventing the wheel"* finally ended up becoming enumext.

Why list environments?

The answer is simple, first I love the beauty of its syntax and many of what I had already written used the enumerate environment or lists created using the enumitem package. In my mind I thought: how complicated could it be to write a package that looked like enumitem? It seemed simple enough, of course I didn't have in mind the mess I was getting into working with list environments, minipage and adding support for the multicol and hyperref packages.
Of course, seeing the final result of the experiment *"reinventing the wheel"* I am quite satisfied.

Why not random questions and other utilities

The *"random"* type questions I love and hate them at the same time, although they simplify a lot the work when creating a multiple choice test, but you lose the beauty of typesetting a document with LaTeX, that is to say the output does not always look as nice as it should, even if they are only alternatives these must follow a certain order when presented either numerical or presentation, that said handling that using *nested lists* is quite complicated so I do not classify to be implemented.

Why has it taken so long?

One of the setbacks, beyond my laziness, was including compatibility with *tagged* PDF. To be honest, it's something I never considered at any point, but I firmly believe that being able to create *accessible documents* provides a great opportunity in the world of mathematics education. From my perspective as a *high school* teacher, beyond theorems and deep mathematics, the use of exercise lists is one of the most common things. Being able to open the way to work in parallel with those who have different abilities is really important and I regret not having looked into this in the past. I hope that enumext serves this purpose and inspires more users and authors to follow this path.

10   References

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

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

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

[4] GONZÁLEZ, PABLO. "scontents - Stores LaTeX contents in memory or files". Available from CTAN, https://www.ctan.org/pkg/scontents, 2024.

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

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

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

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

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

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

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

[12] The LaTeX Project. "The LaTeX 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.

[17] FISCHER, ULRIKE. "tagpdf – LaTeX kernel code for PDF tagging". Available from CTAN, `https://www.ctan.org/pkg/tagpdf`, 2024.

[18] The LaTeX Project. "latex-lab – LaTeX laboratory". Available from CTAN, `https://www.ctan.org/pkg/latex-lab`, 2024.

[19] MITTELBACH, FRANK. "LaTeX's socket management". Available from CTAN, `https://ctan.org/tex-archive/macros/latex/base`, 2024.

11 Change history

v1.2 (ctan), 2025-03-28	– Replace signature (prevent expansion for optional arg).
	– Solve Inconsistent local/global assignment.
v1.1 (ctan), 2024-11-14	– Fixed implementation for `font` and `base-fix` keys.
	– Added new keys for symbol marks.
	– Update and improvements in the internal code.
	– Adjustments in the documentation.
v1.0 (ctan), 2024-11-01	– First public release.

12 Index of Documentation

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

13   Implementation

The most recent publicly released version of enumext is available at CTAN: https://www.ctan.org/pkg/enumext. While general feedback via email is welcomed, specific bugs or feature requests should be reported through the issue tracker: ⟲ https://github.com/pablgonz/enumext/issues.

☙ The documentation presented here is far from professional, it contains a lot of obvious information that to the eye of a TEXpert are superfluous, but, after so many years developing this project is the only way to remember what does what.

13.1   General conventions

Variables containing i, ii, iii and iv are associated by level with the enumext environment, variables containing v are associated with the keyans environment, variables containing vi are associated with the keyanspic environment, variables containing vii are associated with the enumext* environment and variables containing viii are associated with the keyans* environment.

To simplify writing and documentation some variables and functions that are common to the different levels of the environments are described using a capital "X".

The temporary function __enumext_tmp:n is used in different parts of the package code for variable creation or execution of other functions that are grouped into this one.

All variables and functions defined in this package are private and are NOT intended to work or be used by another package or module.

13.2   Initial set up

Start the DocStrip guards.

```
1  ⟨*package⟩
```

Identify the internal prefix (LATEX3 DocStrip convention) for l3doc class.

```
2  ⟨@@=enumext⟩
```

13.3   Declaration of the package

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

```
3  \NeedsTeXFormat{LaTeX2e}[2024-11-01]
```

Now declare the enumext package.

```
4  \ProvidesExplPackage {enumext} {2025-03-28} {1.2} {Enumerate exercise sheets}
```

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

```
5   \hook_gput_code:nnn {begindocument} {enumext}
6     {
7       \IfPackageLoadedTF { multicol }
8         {
9           \msg_info:nnn { enumext } { package-load } { multicol }
10        }
11        {
12          \msg_info:nnn { enumext } { package-not-load } { multicol }
13          \RequirePackage{multicol}[2024-05-23]
14        }
15      \IfPackageLoadedTF { scontents }
16        {
17          \msg_info:nnn { enumext } { package-load } { scontents }
18        }
19        {
20          \msg_info:nnn { enumext } { package-not-load } { scontents }
21          \RequirePackage{scontents}
22        }
23    }
```

13.4   Definition of variables

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

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

```
\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
```

```
24  \int_new:N  \l__enumext_level_int
25  \int_new:N  \l__enumext_level_h_int
26  \int_new:N  \l__enumext_anskey_level_int
27  \int_new:N  \l__enumext_keyans_level_int
28  \int_new:N  \l__enumext_keyans_level_h_int
29  \int_new:N  \l__enumext_keyans_pic_level_int
```

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

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

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

```
30 \bool_new:N \l__enumext_starred_bool
31 \bool_new:N \g__enumext_starred_bool
32 \bool_new:N \l__enumext_starred_first_bool
33 \bool_new:N \l__enumext_standar_bool
34 \bool_new:N \g__enumext_standar_bool
35 \bool_new:N \l__enumext_standar_first_bool
36 \bool_new:N \l__enumext_anskey_env_bool
37 \bool_new:N \l__enumext_keyans_env_bool
38 \tl_new:N   \g__enumext_start_line_tl
39 \tl_new:N   \g__enumext_envir_name_tl
40 \tl_new:N   \l__enumext_envir_name_tl
```

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

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

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

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

```
41 \cs_set_protected:Npn \__enumext_tmp:n #1
42   {
43     \tl_new:c { l__enumext_counter_#1_tl }
44   }
45 \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 (§13.14).

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

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

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

Internal variables used by resume, resume* and series keys (§13.25).

```
57 \int_new:N  \g__enumext_resume_int
58 \int_new:N  \g__enumext_resume_vii_int
59 \tl_new:N    \l__enumext_resume_name_tl
60 \bool_new:N \l__enumext_resume_active_bool
61 \tl_new:N    \g__enumext_standar_series_tl
62 \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 (§13.15) and label (§13.13) keys.

```
63 \dim_new:N \l__enumext_current_widest_dim
64 \tl_new:N  \g__enumext_counter_styles_tl
65 \tl_new:N  \g__enumext_widest_label_tl
66 \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 (§13.18). The variables `\l__enumext_leftmargin_X_dim` and `\l__enumext_itemindent_X_dim` are used and set by the function `\__enumext_calc_hspace:NNNNNNNNNNNN` (§13.38.1).

```
67 \cs_set_protected:Npn \__enumext_tmp:n #1
68   {
69     \bool_new:c { l__enumext_leftmargin_tmp_#1_bool }
70     \dim_new:c  { l__enumext_leftmargin_tmp_#1_dim  }
71     \dim_new:c  { l__enumext_leftmargin_#1_dim      }
72     \dim_new:c  { l__enumext_itemindent_#1_dim      }
73   }
74 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }
```

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

\l__enumext_multicols_above_X_skip
\l__enumext_multicols_below_X_skip
\g__enumext_multicols_right_X_skip
\l__enumext_align_label_pos_X_str

Internal variables used by `columns` key (§13.22) and `align` key (§13.13).

```
75 \cs_set_protected:Npn \__enumext_tmp:n #1
76   {
77     \skip_new:c  { l__enumext_multicols_above_#1_skip }
78     \skip_new:c  { l__enumext_multicols_below_#1_skip }
79     \skip_new:c  { g__enumext_multicols_right_#1_skip }
80     \str_new:c   { l__enumext_align_label_pos_#1_str  }
81   }
82 \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }
```

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

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

Internal variables used by `\miniright` command (§13.23.4) and the keys `mini-right`, `mini-right*`, `mini-env` and `mini-sep` (§13.21, §13.23).

```
83 \int_new:N  \g__enumext_minipage_stat_int
84 \skip_new:N \l__enumext_minipage_temp_skip
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 bool vars `\l__enumext_wrap_label_X_bool` and `\l__enumext_wrap_label_opt_X_bool` are used by `wrap-label` and `wrap-label*` keys (§13.13), the integer `\l__enumext_start_X_int` are used by the `start` and `start*` keys (§13.15), the token list `\l__enumext_fake_item_indent_X_tl` is used by `itemindent` key (§13.18.1), the variables `\l__enumext_label_fill_left_X_tl` and `\l__enumext_label_fill_left_X_tl` are used by the `align` key (§13.13). The boolean vars `\l__enumext_vspace_a_star_X_bool`, `\l__enumext_vspace_b_star_X_bool` are used by `above`, `above*`, `below` and `below*` keys (§13.20).

```
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.*)

The variable `\l__enumext_store_active_bool` setting by `save-ans` key (§13.26.1) 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` (§13.30) 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*` (§13.31).

The variables `\l__enumext_store_current_label_tl` and `\l__enumext_store_current_opt_arg_-tl` save the *current label* and *optional argument* of `\item*` (§13.37) and `\anspic*` (§13.42.2) for the `keyans`, `keyans*` and `keyanspic` environments.

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

```
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.*)

Internal variables used by the command `\setenumext` (§13.48).

```
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.*)

Internal variables used by the `\printkeyans` command (§13.47) and `\foreachkeyans` command (§13.50).

```
122 \tl_new:N  \l__enumext_meta_path_tl
123 \seq_new:N \l__enumext_foreach_print_seq
124 \tl_new:N  \l__enumext_foreach_name_prop_tl
125 \tl_new:N  \l__enumext_foreach_default_keys_tl
```

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

Internal variables used by command `\printkeyans` (§13.47), `show-pos`, `show-ans`, `mark-pos`, `mark-sep` keys (§13.27), `item-sym*` key (§13.35), `save-key` key (§13.27.3) and *"storing structure"*.

```
126 \tl_new:N   \l__enumext_print_keyans_starred_tl
127 \bool_new:N \l__enumext_print_keyans_star_bool
128 \str_new:N  \l__enumext_mark_position_str
129 \str_new:N  \l__enumext_mark_position_v_str
130 \str_new:N  \l__enumext_mark_position_viii_str
131 \dim_new:N  \l__enumext_mark_sep_tmpa_dim
132 \dim_new:N  \l__enumext_mark_sep_tmpb_dim
133 \int_new:N  \l__enumext_show_pos_tmp_int
134 \tl_new:N   \g__enumext_item_symbol_aux_tl
135 \cs_set_protected:Npn \__enumext_tmp:n #1
136   {
137     \tl_new:c   { l__enumext_print_keyans_#1_tl       }
138     \tl_new:c   { l__enumext_store_save_key_#1_tl      }
139     \bool_new:c { l__enumext_store_save_key_#1_bool    }
140     \bool_new:c { l__enumext_store_upper_level_#1_bool }
141   }
142 \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {#1} }
```

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

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

Internal variables used by `keyanspic` environment and `\anspic` command (§13.42.1).

```
143 \seq_new:N  \l__enumext_anspic_args_seq
144 \dim_new:N  \l__enumext_anspic_mini_width_dim
145 \int_new:N  \l__enumext_anspic_above_int
146 \int_new:N  \l__enumext_anspic_below_int
147 \bool_new:N \l__enumext_anspic_label_above_bool
148 \str_new:N  \l__enumext_anspic_mini_pos_str
149 \box_new:N  \l__enumext_anspic_label_box
150 \box_new:N  \l__enumext_anspic_body_box
151 \dim_new:N  \l__enumext_anspic_label_htdp_dim
152 \dim_new:N  \l__enumext_anspic_body_htdp_dim
```

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

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

Internal variables used by *"internal check answer"* mechanism (§13.26.3) used by the `check-ans`, `no-store`, `wrap-ans*` keys and check for starred commands `\item*` in `keyans` and `keyans*` environments and `\anspic*` in `keyanspic` environment.

```
153 \bool_new:N \l__enumext_check_answers_bool
154 \bool_new:N \g__enumext_check_ans_key_bool
155 \tl_new:N   \l__enumext_check_start_line_env_tl
156 \bool_new:N \l__enumext_item_wrap_key_bool
157 \int_new:N  \g__enumext_check_starred_cmd_int
158 \int_new:N  \g__enumext_item_anskey_int
159 \int_new:N  \g__enumext_item_number_int
160 \bool_new:N \l__enumext_item_number_bool
161 \int_new:N  \g__enumext_item_answer_diff_int
```

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

\l__enumext_hyperref_bool
\l__enumext_footnotes_key_bool

The boolean variable `\l__enumext_hyperref_bool` will determine if the `hyperref` package is present or load in memory (§13.7). The boolean variable `\l__enumext_footnotes_key_bool` determine if `hyperref` is load with key `hyperfootnotes`=`true`.

```
162 \bool_new:N \l__enumext_hyperref_bool
163 \bool_new:N \l__enumext_footnotes_key_bool
```

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

\l__enumext_newlabel_arg_one_tl
\l__enumext_newlabel_arg_two_tl
\l__enumext_write_aux_file_tl
\l__enumext_label_copy_X_tl

Internal variables used by `save-ref` key (§13.27). The variables `\l__enumext_label_copy_X_tl` correspond to temporary copies of the ⟨*labels*⟩ defined by level on which operations will be performed.

The variables `\l__enumext_newlabel_arg_one_tl` and `\l__enumext_newlabel_arg_two_tl` will be used to form the arguments passed to the function `\__enumext_newlabel:nn` (§13.7) and the variable `\l__enumext_write_aux_file_tl` will be in charge of executing the writing code in the `.aux` file.

```
164 \tl_new:N \l__enumext_newlabel_arg_one_tl
165 \tl_new:N \l__enumext_newlabel_arg_two_tl
166 \tl_new:N \l__enumext_write_aux_file_tl
167 \cs_set_protected:Npn \__enumext_tmp:n #1
168   {
169     \tl_new:c { l__enumext_label_copy_#1_tl }
170   }
171 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }
```

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

\g__enumext_footnote_standar_int
\g__enumext_footnote_starred_int
\g__enumext_footnote_standar_arg_seq
\g__enumext_footnote_starred_arg_seq
\g__enumext_footnote_standar_int_seq
\g__enumext_footnote_starred_int_seq

Internal variables used for redefinition of `\footnote` (§13.8).

```
172 \int_new:N \g__enumext_footnote_standar_int
173 \int_new:N \g__enumext_footnote_starred_int
174 \seq_new:N \g__enumext_footnote_standar_arg_seq
175 \seq_new:N \g__enumext_footnote_starred_arg_seq
176 \seq_new:N \g__enumext_footnote_standar_int_seq
177 \seq_new:N \g__enumext_footnote_starred_int_seq
```

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

\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

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

```
178 \cs_set_protected:Npn \__enumext_tmp:n #1
179   {
180     \bool_new:c { l__enumext_item_starred_#1_bool    }
181     \int_new:c  { l__enumext_item_column_pos_#1_int  }
182     \int_new:c  { g__enumext_item_count_all_#1_int   }
183     \int_new:c  { l__enumext_joined_item_#1_int      }
```

```
184        \int_new:c  { l__enumext_joined_item_aux_#1_int   }
185        \int_new:c  { l__enumext_tmpa_#1_int               }
186        \dim_new:c  { l__enumext_tmpa_#1_dim               }
187        \box_new:c  { l__enumext_item_text_#1_box          }
188        \dim_new:c  { l__enumext_joined_width_#1_dim       }
189        \dim_new:c  { l__enumext_item_width_#1_dim         }
190        \tl_new:c   { g__enumext_item_symbol_aux_#1_tl     }
191        \str_new:c  { l__enumext_align_label_#1_str        }
192        \bool_new:c { g__enumext_minipage_active_#1_bool   }
193        \box_new:c  { l__enumext_miniright_code_#1_box     }
194        \bool_new:c { g__enumext_minipage_center_#1_bool   }
195        \dim_new:c  { g__enumext_minipage_right_#1_dim     }
196        \skip_new:c { g__enumext_minipage_right_#1_skip    }
197     }
198  \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`.

```
199  \clist_const:Nn \c__enumext_all_envs_clist
200     {
201        {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
202        {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
203     }
```

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

13.5   Some utility functions

`\keys_precompile:neN`   Non-standard kernel variants used by the `\printkeyans` command (§13.47) and `\foreachkeyans` command
`\seq_use:NV`   (§13.50).

```
204  \cs_generate_variant:Nn \keys_precompile:nnN { neN }
205  \cs_generate_variant:Nn \seq_use:Nn { NV }
```

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

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

```
206  \cs_new_protected:Npn \__enumext_at_begin_document:n #1
207     {
208        \hook_gput_code:nnn {begindocument} {enumext} { #1 }
209     }
```

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

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

```
210  \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
211     {
212        \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
213     }
214  \cs_new_protected:Npn \__enumext_before_env:nn #1 #2
215     {
216        \hook_gput_code:nnn {env/#1/before} {enumext} {#2}
217     }
```

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

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

```
218  \cs_new:Nn \__enumext_level:
219     {
220        \int_to_roman:n { \l__enumext_level_int }
221     }
```

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

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

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

```
222  \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
223    {
224      \regex_match:nnTF { ^[\+\-]?[\d]+$ } {#1} % $
225        { \prg_return_true: }
226        { \prg_return_false: }
227    }
```

(*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.

```
228  \cs_new_protected:Nn \__enumext_regex_counter_style:
229    {
230      \tl_map_inline:Nn \c__enumext_counter_style_tl
231        {
232          \regex_replace_once:nnN { \c{##1}\* }
233            { \c{##1}\cB{\u{l__enumext_ref_the_count_tl}\cE} } \l__enumext_ref_key_arg_tl
234        }
235    }
```

(*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.

```
236  \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
237    {
238      * ~ #2
239      \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
240        = ~ \use:c { #1_use:c } { l__enumext_#2_#3_#1 } \\
241    }
```

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

`\__enumext_unskip_unkern:`

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

```
242  \cs_new_protected:Nn \__enumext_unskip_unkern:
243    {
244      \int_case:nnT { \lastnodetype }
245        {
246          { 11 }{ \unskip }
247          { 12 }{ \unkern }
248        }
249    }
```

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

13.5.1   Utilities for environments and levels

`\__enumext_is_not_nested:`
`\__enumext_is_on_first_level:`

The function `\__enumext_is_not_nested:` set the variables `\g__enumext_standar_bool` and `\g__-enumext_starred_bool` to *"true"* only if the environments `enumext` and `enumext*` are NOT nested in each other and save the environment name in `\l__enumext_envir_name_tl`.

```
250  \cs_new_protected:Nn \__enumext_is_not_nested:
251    {
252      \str_case:en { \@currenvir }
253        {
254          {enumext}
255            {
256              \tl_set:Nn \l__enumext_envir_name_tl { enumext }
257              \bool_lazy_and:nnT
258                { \bool_not_p:n { \g__enumext_standar_bool } }
259                { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
260                {
261                  \bool_gset_true:N \g__enumext_standar_bool
262                }
263            }
264          {enumext*}
```

```
265          {
266            \tl_set:Nn \l__enumext_envir_name_tl { enumext* }
267            \bool_lazy_and:nnT
268              { \bool_not_p:n { \g__enumext_starred_bool } }
269              { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
270              {
271                \bool_gset_true:N \g__enumext_starred_bool
272              }
273          }
274        }
275      }
```

The function `\__enumext_is_on_first_level:` will set the variables `\l__enumext_standar_first_-bool` (§13.26.1), `\l__enumext_starred_first_bool` (§13.26.1) and `\l__enumext_anskey_env_bool` (§13.31) 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.

```
276  \cs_new_protected:Nn \__enumext_is_on_first_level:
277    {
278      \bool_lazy_all:nT
279        {
280          { \bool_if_p:N \g__enumext_standar_bool }
281          { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
282          { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
283        }
284        {
285          \bool_set_true:N \l__enumext_standar_first_bool
286          \bool_set_true:N \l__enumext_anskey_env_bool
287          \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
288          \tl_gset:Ne \g__enumext_start_line_tl
289            {
290              on ~ line ~ \exp_not:V \inputlineno
291            }
292        }
293      \bool_lazy_all:nT
294        {
295          { \bool_if_p:N \g__enumext_starred_bool }
296          { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
297          { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
298        }
299        {
300          \bool_set_true:N \l__enumext_starred_first_bool
301          \bool_set_true:N \l__enumext_anskey_env_bool
302          \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
303          \tl_gset:Ne \g__enumext_start_line_tl
304            {
305              on ~ line ~ \exp_not:V \inputlineno
306            }
307        }
308    }
```

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

`\__enumext_keyans_name_and_start:`   The function `\__enumext_keyans_name_and_start:` will save the start line number and name of the environments keyans, keyans* and keyanspic in the variables `\l__enumext_check_start_line_env_-tl` and `\l__enumext_envir_name_tl` to use in the `\__enumext_check_starred_cmd:n` function.

```
309  \cs_new_protected:Nn \__enumext_keyans_name_and_start:
310    {
311      \str_case:en { \@currenvir }
312        {
313          {keyans}
314            {
315              \tl_set:Nn \l__enumext_envir_name_tl { keyans }
316              \tl_set:Ne \l__enumext_check_start_line_env_tl
317                {
318                  in ~ 'keyans' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
319                }
320            }
321          {keyans*}
322            {
```

```
323            \tl_set:Nn \l__enumext_envir_name_tl { keyans* }
324            \tl_set:Ne \l__enumext_check_start_line_env_tl
325              {
326                in ~ 'keyans*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
327              }
328          }
329        {keyanspic}
330          {
331            \tl_set:Nn \l__enumext_envir_name_tl { keyanspic }
332            \tl_set:Ne \l__enumext_check_start_line_env_tl
333              {
334                in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
335              }
336          }
337      }
338    }
```

(*End of definition for* __enumext_keyans_name_and_start:.)

13.5.2   Utilities for log and terminal

__enumext_reset_global_vars:
__enumext_reset_global_int:
__enumext_reset_global_bool:
__enumext_reset_global_tl:

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

```
339 \cs_new_protected:Nn \__enumext_reset_global_vars:
340   {
341     \__enumext_reset_global_int:
342     \__enumext_reset_global_bool:
343     \__enumext_reset_global_tl:
344   }
345 \cs_new_protected:Nn \__enumext_reset_global_int:
346   {
347     \int_gzero:N \g__enumext_item_number_int
348     \int_gzero:N \g__enumext_item_anskey_int
349     \int_gzero:N \g__enumext_item_answer_diff_int
350   }
351 \cs_new_protected:Nn \__enumext_reset_global_bool:
352   {
353     \bool_gset_false:N \g__enumext_check_ans_key_bool
354     \bool_gset_false:N \g__enumext_standar_bool
355     \bool_gset_false:N \g__enumext_starred_bool
356   }
357 \cs_new_protected:Nn \__enumext_reset_global_tl:
358   {
359     \tl_gclear:N \g__enumext_store_name_tl
360     \tl_gclear:N \g__enumext_start_line_tl
361     \tl_gclear:N \g__enumext_envir_name_tl
362   }
```

(*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.

```
363 \cs_new_protected:Nn \__enumext_log_global_vars:
364   {
365     \msg_log:nneeee { enumext } { prop-seq-int-hook }
366       { \g__enumext_store_name_tl }
367       { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
368       { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
369       { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
370   }
```

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.

```
371 \cs_new_protected:Nn \__enumext_log_answer_vars:
372   {
373     \msg_log:nneee { enumext } { item-answer-hook }
374       { \int_use:N \g__enumext_item_number_int }
375       { \int_use:N \g__enumext_item_anskey_int }
376       { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int} }
377   }
```

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

13.6 Copying `list` and `minipage` environments

The `list` environment provided by LATEX has the following plain form:

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

And `minipage` environment provided by LATEX has the following (simplified) plain form:

```
\minipage[⟨pos⟩][⟨height⟩][⟨inner-pos⟩]{⟨width⟩}
  ⟨internal implement⟩
\endminipage
```

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

🔹 For compatibility with *tagged* PDF we should use `\NewCommandCopy` and not `\cs_new_eq:NN` for `\item`. When *tagged* PDF is active `\item` is redefined using `ltcmd` (see `latex-lab-block`[18]).

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

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

```
378 \__enumext_at_begin_document:n
379   {
380     \cs_new_eq:NN    \__enumext_start_list:nn \list
381     \cs_new_eq:NN    \__enumext_stop_list: \endlist
382     \NewCommandCopy \__enumext_item_std:w \item
383   }
```

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

```
384 \__enumext_at_begin_document:n
385   {
386     \cs_new_eq:NN \__enumext_minipage:w \minipage
387     \cs_new_eq:NN \__enumext_endminipage: \endminipage
388   }
```

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

13.7 Compatibility with hyperref and footnotehyper

`\__enumext_after_hyperref:`
`\__enumext_hypertarget:nn`
`\__enumext_phantomsection:`

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

```
389 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
390 \hook_gset_rule:nnnn { begindocument } { enumext } { after } { hyperref }
```

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

```
391 \cs_new_protected:Nn \__enumext_after_hyperref:
392   {
393     \IfPackageLoadedTF { hyperref }
394       {
395         \msg_info:nnn { enumext } { package-load } { hyperref }
396         \bool_set_true:N \l__enumext_hyperref_bool
397         \IfHyperBoolean{hyperfootnotes}
398           {
399             \bool_set_true:N \l__enumext_footnotes_key_bool
400           }
401           {  }
402       }
403       {  }
```

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`.

```
404     \bool_if:NT \l__enumext_footnotes_key_bool
405       {
406         \IfPackageLoadedTF { footnotehyper }
407           {
408             \msg_info:nnn { enumext } { package-load } { footnotehyper }
409           }
410           {
```

```
411        \bool_set_false:N \l__enumext_footnotes_key_bool
412      }
413    }
```

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.

```
414    \bool_if:NTF \l__enumext_hyperref_bool
415      {
416        \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
417        \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
418      }
419      {
420        \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
421        \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
422      }
423    }
```

(*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.

```
424 \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
425    {
426      \protected@write \@auxout { }
427        {
428          \token_to_str:N \newlabel {#1}
429            {
430              {#2}
431              \bool_if:NT \l__enumext_hyperref_bool
432                { { \thepage } {#2} {#1} }
433              { }
434            }
435        }
436      \__enumext_hypertarget:nn {#1} { }
437      \__enumext_phantomsection:
438    }
```

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

13.8   Internal redefining \footnote **command**

To keep the correct numbering of \footnote and to make it work correctly in the enumext* and keyans* environments and mini-env key it is necessary to redefine the \footnote command. This implementation is adapted from the answer given by Clea F. Rees (@cfr) in footnotes in boxes compatible with hyperref.

__enumext_footnotetext:nn   Redefinition of the \footnote command using \footnotetext and \footnotemark for the mini-env key
__enumext_renew_footnote: in the enumext and keyans environments.
__enumext_print_footnote:
__enumext_renew_footnote_mini:
__enumext_print_footnote_mini:

```
439 \cs_new_protected:Nn \__enumext_footnotetext:nn
440    {
441      \footnotetext[#1]{#2}
442    }
443 \cs_new_protected:Nn \__enumext_renew_footnote:
444    {
445      \RenewDocumentCommand \footnote { o +m }
446        {
447          \tl_if_novalue:nTF {##1}
448            {
449              \stepcounter{footnote}
450              \int_gset_eq:Nc \g__enumext_footnote_standar_int { c@footnote }
451            }
452            {
453              \int_gset:Nn \g__enumext_footnote_standar_int { ##1 }
454            }
455          \footnotemark [ \g__enumext_footnote_standar_int ]
456          \seq_gput_right:Nn \g__enumext_footnote_standar_arg_seq { ##2 }
457          \seq_gput_right:NV
```

```
458          \g__enumext_footnote_standar_int_seq \g__enumext_footnote_standar_int
459        }
460    }
461  \cs_new_protected:Nn \__enumext_print_footnote:
462    {
463      \seq_if_empty:NF \g__enumext_footnote_standar_int_seq
464        {
465          \seq_map_pairwise_function:NNN
466            \g__enumext_footnote_standar_int_seq
467            \g__enumext_footnote_standar_arg_seq
468            \__enumext_footnotetext:nn
469        }
470      \seq_gclear:N \g__enumext_footnote_standar_arg_seq
471      \seq_gclear:N \g__enumext_footnote_standar_int_seq
472    }
```

The enumext* and keyans* environments are implemented using minipage so we must also redefine \footnote to keep these numbering as if it were part of the document.

```
473  \cs_new_protected:Nn \__enumext_renew_footnote_mini:
474    {
475      \RenewDocumentCommand \footnote { o +m }
476        {
477          \tl_if_novalue:nTF {##1}
478            {
479              \stepcounter{footnote}
480              \int_gset_eq:Nc \g__enumext_footnote_starred_int { c@footnote }
481            }
482            {
483              \int_gset:Nn \g__enumext_footnote_starred_int { ##1 }
484            }
485          \footnotemark [ \g__enumext_footnote_starred_int ]
486          \seq_gput_right:Nn \g__enumext_footnote_starred_arg_seq { ##2 }
487          \seq_gput_right:NV
488            \g__enumext_footnote_starred_int_seq \g__enumext_footnote_starred_int
489        }
490    }
491  \cs_new_protected:Nn \__enumext_print_footnote_mini:
492    {
493      \seq_if_empty:NF \g__enumext_footnote_starred_int_seq
494        {
495          \seq_map_pairwise_function:NNN
496            \g__enumext_footnote_starred_int_seq
497            \g__enumext_footnote_starred_arg_seq
498            \__enumext_footnotetext:nn
499        }
500      \seq_gclear:N \g__enumext_footnote_starred_arg_seq
501      \seq_gclear:N \g__enumext_footnote_starred_int_seq
502    }
```

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

__enumext_renew_footnote_standar:
__enumext_print_footnote_standar:
__enumext_renew_footnote_starred:
__enumext_print_footnote_starred:

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

```
503  \cs_new_protected:Nn \__enumext_renew_footnote_standar:
504    {
505      \bool_if:NT \g__enumext_standar_bool
506        {
507          \IfDocumentMetadataTF
508            {
509              \__enumext_renew_footnote:
510            }
511            {
512              \bool_if:NF \l__enumext_footnotes_key_bool
513                {
514                  \__enumext_renew_footnote:
515                }
516            }
517        }
518    }
519  \cs_new_protected:Nn \__enumext_print_footnote_standar:
```

```
520    {
521      \bool_if:NT \g__enumext_standar_bool
522        {
523          \IfDocumentMetadataTF
524            {
525              \__enumext_print_footnote:
526            }
527            {
528              \bool_if:NF \l__enumext_footnotes_key_bool
529                {
530                  \__enumext_print_footnote:
531                }
532            }
533        }
534    }
```

We encapsulate the redefinition of \footnote to pass it to the enumext* and keyans* environments. We will run the redefinition when *tagged* PDF is active or when the footnotehyper package is not loaded.

```
535  \cs_new_protected:Nn \__enumext_renew_footnote_starred:
536    {
537      \IfDocumentMetadataTF
538        {
539          \__enumext_renew_footnote_mini:
540        }
541        {
542          \bool_if:NF \l__enumext_footnotes_key_bool
543            {
544              \__enumext_renew_footnote_mini:
545            }
546        }
547    }
548  \cs_new_protected:Nn \__enumext_print_footnote_starred:
549    {
550      \IfDocumentMetadataTF
551        {
552          \__enumext_print_footnote_mini:
553        }
554        {
555          \bool_if:NF \l__enumext_footnotes_key_bool
556            {
557              \__enumext_print_footnote_mini:
558            }
559        }
560    }
```

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

```
561  \__enumext_after_env:nn { enumext* }
562    {
563      \__enumext_print_footnote_starred:
564    }
565  \__enumext_after_env:nn { keyans* }
566    {
567      \__enumext_print_footnote_starred:
568    }
```

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

13.9   The internal minipage environment

__enumext_internal_mini_page:
__enumext_mini_env*

The function __enumext_internal_mini_page: creates a internal __enumext_mini_page environment (*custom version* of minipage) setting the \if@minipage switch to *"false"* to allow spaces at the *"above"* of the environment, plus we will add \skip_vertical:N \c_zero_skip to maintain alignment on *"top"* in the first part and \skip_vertical:N \c_zero_skip in the second part to allow spaces *"below"*. This environment will be used internally by the mini-env key, it is NOT documented in the user interface and is for internal use only. Within this environment we redefine \footnote to make them look the same as if they were elsewhere in the document. This function is passed to the function __enumext_safe_exec: in the enumext environment definition (§13.39) and __enumext_safe_exec_vii: in the enumext* environment definition (§13.44)

```
569  \cs_new_protected:Nn \__enumext_internal_mini_page:
570    {
571      \int_compare:nNnT { \l__enumext_level_int } = { 0 }
```

```
572        {
573            \DeclareDocumentEnvironment{__enumext_mini_page}{ m }
574              {
575                \__enumext_renew_footnote_standar:
576                \__enumext_minipage:w [ t ] { ##1 }
577                  \legacy_if_gset_false:n { @minipage }
578                  \skip_vertical:N \c_zero_skip
579              }
580              {
581                  \skip_vertical:N \c_zero_skip
582                \__enumext_endminipage:
583                \__enumext_print_footnote_standar:
584              }
585          }
586      }
```

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

13.10   Definition of public dimension

The package enumext only provides a single public dimension `\itemwidth` and is intended for user convenience only and is not for internal use as such. This dimension is set in all environments and is only used by the `wrap-ans` key at its default value.

```
587  \dim_zero_new:N \itemwidth
```

13.11   Definition of counters

`\__enumext_define_counters:Nn`
`enumXi`
`enumXii`
`enumXiii`
`enumXiv`
`enumXv`
`enumXvi`
`enumXvii`
`enumXviii`

To create the necessary *"counters"* we must first make sure that they are not already defined by the user or a package such as enumitem, otherwise a error will be returned and the package loading will be aborted. The arguments taken by the function are:

#1 :   A token list `\l__enumext_counter_X_tl` for *"store"* the counter's name.

#2 :   The counter's name.

```
588  \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
589    {
590      \cs_if_exist:cTF { c@ #2 }
591        { \msg_fatal:nnn { enumext } { counters }{ #2 } }
592        {
593          \tl_set:Nn #1 { #2 }
594          \newcounter { #2 }
595        }
596    }
```

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.

```
597  \__enumext_define_counters:Nn \l__enumext_counter_i_tl     { enumXi     }
598  \__enumext_define_counters:Nn \l__enumext_counter_ii_tl    { enumXii    }
599  \__enumext_define_counters:Nn \l__enumext_counter_iii_tl   { enumXiii   }
600  \__enumext_define_counters:Nn \l__enumext_counter_iv_tl    { enumXiv    }
601  \__enumext_define_counters:Nn \l__enumext_counter_v_tl     { enumXv     }
602  \__enumext_define_counters:Nn \l__enumext_counter_vi_tl    { enumXvi    }
603  \__enumext_define_counters:Nn \l__enumext_counter_vii_tl   { enumXvii   }
604  \__enumext_define_counters:Nn \l__enumext_counter_viii_tl  { enumXviii  }
```

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

13.12   Definition of labels

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

`\__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, enumext*, keyans and keyans* environments, so it is necessary to get a default value for `labelwidth` from these ⟨*labels*⟩ at the same time.

```
605  \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
606    {
607      \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
608      \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
609    }
610  \__enumext_register_counter_style:Nn \arabic { 0 }
611  \__enumext_register_counter_style:Nn \Alph   { M }
612  \__enumext_register_counter_style:Nn \alph   { m }
```

```
613 \__enumext_register_counter_style:Nn \Roman  { VIII }
614 \__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.

```
615 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
616   {
617     \hbox_set:Nn \l__enumext_label_width_by_box {#2}
618     \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
619   }
620 \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.

```
621 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
622   {
623     \tl_clear_new:N #1
624     \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
625     \tl_gset_eq:NN \g__enumext_widest_label_tl #1
626     \tl_map_inline:Nn \g__enumext_counter_styles_tl
627       {
628         \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
629         \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
630           { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
631       }
632     \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
633       { \tl_use:N \g__enumext_widest_label_tl }
634     \tl_set_eq:cN { the #2 } #1
635   }
636 \cs_generate_variant:Nn \__enumext_label_style:Nnn { cvn }
```

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

13.13 Setting keys associated with label

When *tagged* PDF is active \makelabel is redefined using \makebox to work correctly (§13.34). From the user side it is convenient to have a key that allows using this redefinition with \makebox without having \IfDocumentMetadataTF active.

mode-box

We define the key mode-box only for the *"first level"* of enumext and enumext* environments.

```
637 \cs_set_protected:Npn \__enumext_tmp:n #1
638   {
639     \keys_define:nn { enumext /  #1 }
640       {
641         mode-box .bool_set:N = \l__enumext_mode_box_bool,
642         mode-box .initial:n  = false,
643         mode-box .value_forbidden:n = true,
644       }
645   }
646 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

(*End of definition for* mode-box*.*)

font
labelsep
labelwidth
wrap-label
wrap-label*

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

```
647 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
648   {
649     \keys_define:nn { enumext / #1 }
650       {
651         font        .tl_set:c   = { l__enumext_label_font_style_#2_tl },
652         font        .value_required:n = true,
653         labelsep    .dim_set:c  = { l__enumext_labelsep_#2_dim },
654         labelsep    .initial:n  = {0.3333em},
655         labelsep    .value_required:n = true,
```

```
656          labelwidth  .dim_set:c  = { l__enumext_labelwidth_#2_dim },
657          labelwidth  .value_required:n = true,
658          wrap-label  .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
659          wrap-label  .initial:n  = {##1},
660          wrap-label  .value_required:n = true,
661          wrap-label* .code:n = {
662                          \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
663                          \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
664                        },
665          wrap-label* .value_required:n = true,
666        }
667    }
668  \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

align The `align` key is implemented differently for *"starred"* and *"non starred"* environments. For compatibility with *tagged* PDF we must set `\l__enumext_align_label_pos_X_str`.

```
669  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
670    {
671      \keys_define:nn { enumext / #1 }
672        {
673          align .choice:,
674          align / left    .code:n =
675                            {
676                              \tl_clear:c { l__enumext_label_fill_left_#2_tl  }
677                              \tl_set:cn  { l__enumext_label_fill_right_#2_tl } { \hfill }
678                              \str_set:cn { l__enumext_align_label_pos_#2_str } { l }
679                            },
680          align / right   .code:n =
681                            {
682                              \tl_set:cn  { l__enumext_label_fill_left_#2_tl  } { \hfill }
683                              \tl_clear:c { l__enumext_label_fill_right_#2_tl }
684                              \str_set:cn { l__enumext_align_label_pos_#2_str } { r }
685                            },
686          align / center  .code:n =
687                            {
688                              \tl_set:cn { l__enumext_label_fill_left_#2_tl  } { \hfill }
689                              \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
690                              \str_set:cn { l__enumext_align_label_pos_#2_str } { c }
691                            },
692          align / unknown .code:n =
693                            \msg_error:nneee { enumext } { unknown-choice }
694                            { align } { left, ~ right, ~  center } { \exp_not:n {##1} },
695          align .initial:n  = left,
696          align .value_required:n  = true,
697        }
698    }
699  \clist_map_inline:nn
700    {
701      {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
702    }
703    { \__enumext_tmp:nn #1 }

704  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
705    {
706      \keys_define:nn { enumext / #1 }
707        {
708          align .choice:,
709          align / left    .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
710          align / right   .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
711          align / center  .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
712          align / unknown .code:n =
713                            \msg_error:nneee { enumext } { unknown-choice }
714                            { align } { left, ~ right, ~  center } { \exp_not:n {##1} },
715          align .initial:n = left,
716          align .value_required:n  = true,
717        }
718    }
719  \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }
```

(*End of definition for align.*)

13.14 Setting `label` and `ref` keys

The implementation of the keys `label` and `ref` are part of the core of the package enumext, here the default values for ⟨*label*⟩, the value of the variables \l__enumext_label_X_tl, the default values for \labelwidth and the *"label and ref"* system.

13.14.1 Define and set `label` and `ref` keys for `enumext` environment

label
ref
\l__enumext_label_i_tl
\l__enumext_label_ii_tl
\l__enumext_label_iii_tl
\l__enumext_label_iv_tl

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

```
720  \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
721    {
722      \keys_define:nn { enumext / #1 }
723        {
724          label .code:n    = {
725                             \__enumext_label_style:cvn { l__enumext_label_#2_tl }
726                             { l__enumext_counter_#2_tl } {##1}
727                             \dim_set_eq:cN  { l__enumext_labelwidth_#2_dim }
728                             \l__enumext_current_widest_dim
729                           },
730          label .initial:n = #3,
731          label .value_required:n = true,
732          ref    .code:n    = \__enumext_standar_ref:n {##1},
733          ref    .value_required:n = true,
734        }
735    }
736  \__enumext_tmp:nnn { level-1 } {   i } { \arabic*.}
737  \__enumext_tmp:nnn { level-2 } {  ii } { (\alph*) }
738  \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
739  \__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.

```
740  \cs_new_protected:Npn \__enumext_standar_ref:n #1
741    {
742      \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
743      \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
744        {
745          \msg_error:nnn { enumext } { key-ref-empty } { enumext }
746        }
747        {
748          \tl_set_eq:Nc
749            \l__enumext_ref_the_count_tl { l__enumext_counter_ \__enumext_level: _tl }
750          \__enumext_regex_counter_style:
751          \tl_set_eq:Nc
752            \l__enumext_ref_the_count_tl { l__enumext_the_counter_ \__enumext_level: _tl }
753          \tl_put_right:ce { l__enumext_renew_the_count_ \__enumext_level: _tl }
754            {
755              \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V \l__
756            }
757        }
758    }
```

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

```
759  \cs_new_protected:Nn \__enumext_standar_ref:
760    {
761      \tl_if_empty:cF { l__enumext_renew_the_count_ \__enumext_level: _tl }
762        {
763          \tl_use:c { l__enumext_renew_the_count_ \__enumext_level: _tl }
764        }
765    }
```

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

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

label
ref
\l__enumext_label_vii_tl
\l__enumext_label_viii_tl

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

```
766 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
767   {
768     \keys_define:nn { enumext / #1 }
769       {
770         label .code:n    = {
771                             \__enumext_label_style:cvn { l__enumext_label_#2_tl }
772                               { l__enumext_counter_#2_tl } {##1}
773                             \dim_set_eq:cN  { l__enumext_labelwidth_#2_dim }
774                               \l__enumext_current_widest_dim
775                           },
776         label .initial:n = #3,
777         label .value_required:n = true,
778         ref   .code:n    = \__enumext_starred_ref:n {##1},
779         ref   .value_required:n = true,
780       }
781   }
782 \__enumext_tmp:nnn { enumext* } {  vii } { \arabic*.}
783 \__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.

```
784 \cs_new_protected:Npn \__enumext_starred_ref:n #1
785   {
786     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
787     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
788       {
789         \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
790           {
791             \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
792           }
793           {
794             \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
795             \__enumext_regex_counter_style:
796             \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
797             \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
798               {
799                 \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V
800               }
801           }
802       }
803     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
804       {
805         \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
806           {
807             \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
808           }
809           {
810             \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
811             \__enumext_regex_counter_style:
812             \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
813             \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
814               {
815                 \exp_not:N \renewcommand  { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V
816               }
817           }
818       }
819   }
```

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.

```
820 \cs_new_protected:Nn \__enumext_starred_ref:
821   {
822     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
823       {
824         \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
825           {
826             \tl_use:N \l__enumext_renew_the_count_vii_tl
```

```
827                    }
828              }
829      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
830          {
831              \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
832                  {
833                      \tl_use:N \l__enumext_renew_the_count_viii_tl
834                  }
835          }
836      }
```

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

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

label

ref

\l__enumext_label_v_tl

\l__enumext_label_vi_tl

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

```
837  \keys_define:nn { enumext / keyans }
838      {
839          label .code:n    = {
840                                  \__enumext_label_style:cvn { l__enumext_label_v_tl }
841                                      { l__enumext_counter_v_tl } {#1}
842                                  \__enumext_label_style:cvn { l__enumext_label_vi_tl }
843                                      { l__enumext_counter_vi_tl } {#1}
844                                  \dim_set_eq:NN
845                                      \l__enumext_labelwidth_v_dim \l__enumext_current_widest_dim
846                              },
847          label .initial:n = \Alph*),
848          label .value_required:n = true,
849          ref   .code:n    = \__enumext_keyans_ref:n {#1},
850          ref   .value_required:n = true,
851      }
```

(*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`.

```
852  \cs_new_protected:Npn \__enumext_keyans_ref:n #1
853      {
854          \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
855          \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
856              {
857                  \msg_error:nnn { enumext } { key-ref-empty } { keyans }
858              }
859              {
860                  \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
861                  \__enumext_regex_counter_style:
862                  \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
863                  \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
864                      {
865                          \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V \l__
866                      }
867              }
868      }
```

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

```
869  \cs_new_protected:Nn \__enumext_keyans_ref:
870      {
871          \tl_if_empty:NF \l__enumext_renew_the_count_v_tl
872              {
873                  \tl_use:N \l__enumext_renew_the_count_v_tl
874              }
875      }
```

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

13.15   Setting start, start* and widest keys

__enumext_start_from:NNn
__enumext_start_from:ccn
__enumext_start_from:cce

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

#1 : \l__enumext_label_X_tl
#2 : \l__enumext_start_X_int
#3 : ⟨integer or string⟩

The first argument of this function are the *"counter style"* set by label key, the second argument is returned by the function, the third argument can be an ⟨integer⟩ or ⟨string⟩ of the form \Alph, \alph, \Roman or \roman. This effectively allows start=A or start=1 to be used.

```
876 \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
877   {
878     \__enumext_if_is_int:nTF { #3 }
879       {
880         \int_set:Nn #2 {#3}
881       }
882       {
883         \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
884           { \int_set:Nn #2 { \int_from_alph:n {#3} } }
885         \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
886           { \int_set:Nn #2  { \int_from_roman:n {#3} } }
887       }
888   }
889 \cs_generate_variant:Nn \__enumext_start_from:NNn { ccn, cce }
```

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

__enumext_widest_from:nNNn
__enumext_widest_from:nccn

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

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

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

```
890 \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
891   {
892     \__enumext_if_is_int:nTF {#4}
893       {
894         \setcounter{enumX#1} { #4 }
895       }
896       {
897         \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
898           { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
899         \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
900           { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
901       }
902     \__enumext_label_width_by_box:cv
903       { l__enumext_labelwidth_#1_dim } { l__enumext_label_#1_tl }
904   }
905 \cs_generate_variant:Nn \__enumext_widest_from:nNNn { nccn }
```

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

start
start*
widest

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

```
906 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
907   {
908     \keys_define:nn { enumext / #1 }
909       {
910         start* .code:n    = {
911                             \__enumext_start_from:ccn
912                               { l__enumext_label_#2_tl }
913                               { l__enumext_start_#2_int } {##1}
914                             },
915         start* .value_required:n = true,
916         start  .code:n    = {
917                             \__enumext_start_from:cce
918                               { l__enumext_label_#2_tl }
919                               { l__enumext_start_#2_int } { \int_eval:n {##1} }
```

```
920                                    },
921          start  .initial:n = 1,
922          start  .value_required:n = true,
923          widest .code:n     = {
924                                  \__enumext_widest_from:nccn {#2}
925                                      { l__enumext_label_#2_tl }
926                                      { l__enumext_labelwidth_#2_dim } {##1}
927                                  },
928          widest .value_required:n = true,
929        }
930      }
931 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

13.16 Setting keys for 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.

```
932 \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
933   {
934     \keys_define:nn { enumext / #1 }
935       {
936         topsep    .skip_set:c = { l__enumext_topsep_#2_skip },
937         topsep    .initial:n  = {#3},
938         topsep    .value_required:n = true,
939         partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
940         partopsep .initial:n  = {#4},
941         partopsep .value_required:n = true,
942         parsep    .skip_set:c = { l__enumext_parsep_#2_skip },
943         parsep    .initial:n  = {#5},
944         parsep    .value_required:n = true,
945         itemsep   .skip_set:c = { l__enumext_itemsep_#2_skip },
946         itemsep   .initial:n  = {#6},
947         itemsep   .value_required:n = true,
948         noitemsep .meta:n     = { itemsep = 0pt, parsep = 0pt },
949         noitemsep .value_forbidden:n = true,
950         nosep     .meta:n     = {
951                                   itemsep = 0pt, parsep= 0pt,
952                                   topsep = 0pt, partopsep = 0pt,
953                                 },
954         nosep     .value_forbidden:n = true,
955       }
956   }
```

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

```
957 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
958   { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
959   { 4.0pt plus 2.0pt minus 1.0pt }
960 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
961   { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
962   { 2.0pt plus 1.0pt minus 1.0pt }
963 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
964   { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
965 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
966   { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
967 \__enumext_tmp:nnnnnn { keyans  } { v }{ 4.0pt plus 2.0pt minus 1.0pt }
968   { 2.0pt plus 1.0pt minus 1.0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
969   { 2.0pt plus 1.0pt minus 1.0pt }
970 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
971   { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
972   { 4.0pt plus 2.0pt minus 1.0pt }
973 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
974   { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
975   { 2.0pt plus 1.0pt minus 1.0pt }
```

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

13.17 Setting base-fix key

When nesting starting right after \item (without material between them) there is a problem with the alignment of the *baseline* between the two environments. One way to get around this problem is to place

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

base-fix

__enumext_nested_base_line_fix:

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

```
976 \keys_define:nn { enumext /  level-1 }
977   {
978     base-fix .bool_set:N = \l__enumext_base_line_fix_bool,
979     base-fix .initial:n  = false,
980     base-fix .value_forbidden:n = true,
981   }
```

The function `\__enumext_nested_base_line_fix:` passed to the `\__enumext_parse_keys:n` function in the definition of the enumext environment (§13.39) will be responsible for applying the *baseline correction* and adjusting the ⟨*keys*⟩ for the enumext environment and the `\printkeyans` with *starred argument* '`*`' (§13.47).

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

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

```
982 \cs_new_protected:Nn \__enumext_nested_base_line_fix:
983   {
984     \bool_lazy_all:nT
985       {
986         { \bool_if_p:N \l__enumext_starred_first_bool }
987         { \bool_if_p:N \l__enumext_base_line_fix_bool }
988         { \bool_not_p:n { \l__enumext_print_keyans_star_bool } }
989       }
990       {
991         \mode_leave_vertical:
992         \vspace { -\dim_eval:n { \baselineskip + \parsep } }
993         \keys_set:nn { enumext / level-1 }
994           {
995             topsep = 0pt, above = 0pt, above* = 0pt,
996           }
997       }
```

When we are running the `\printkeyans` command with the *starred argument* '`*`' the variable `\l__-enumext_print_keyans_star_bool` is true and we can run a simplified version of `\vspace` using `\skip_vertical:n`.

```
998     \bool_lazy_and:nnT
999       { \bool_if_p:N \l__enumext_starred_first_bool }
1000      { \bool_if_p:N \l__enumext_print_keyans_star_bool }
1001      {
1002        \mode_leave_vertical:
1003        \skip_vertical:n { -\baselineskip }
1004        \skip_vertical:N \c_zero_skip
1005        \keys_set:nn { enumext / level-1 }
1006          {
1007            topsep = 0pt, above = 0pt, above* = 0pt,
1008          }
1009      }
1010    \bool_set_false:N \l__enumext_base_line_fix_bool
1011  }
```

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

13.18   Setting keys for horizontal spaces

itemindent

rightmargin

listparindent

list-offset

list-indent

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

```
1012 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1013   {
1014     \keys_define:nn { enumext / #1 }
1015       {
1016         itemindent    .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
1017         itemindent    .value_required:n = true,
1018         rightmargin   .dim_set:c = { l__enumext_rightmargin_#2_dim },
1019         rightmargin   .value_required:n = true,
1020         listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
```

```
1021        listparindent .value_required:n = true,
1022        list-offset    .dim_set:c = { l__enumext_listoffset_#2_dim },
1023        list-offset    .value_required:n = true,
1024        list-indent    .code:n      =
1025                         \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
1026                         \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
1027        list-indent    .value_required:n = true,
1028      }
1029    }
1030  \clist_map_inline:nn
1031    {
1032      {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
1033    }
1034    { \__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.

```
1035  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1036    {
1037      \keys_define:nn { enumext / #1 }
1038        {
1039          itemindent     .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
1040          itemindent     .value_required:n = true,
1041          rightmargin    .dim_set:c = { l__enumext_rightmargin_#2_dim },
1042          rightmargin    .value_required:n = true,
1043          listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
1044          listparindent .value_required:n = true,
1045          list-offset    .dim_set:c = { l__enumext_listoffset_#2_dim },
1046          list-offset    .value_required:n = true,
1047          list-indent    .meta:n     = { list-offset = ##1 },
1048          list-indent    .value_required:n = true,
1049        }
1050    }
1051  \clist_map_inline:nn
1052    {
1053      {enumext*}{vii}, {keyans*}{viii}
1054    }
1055    { \__enumext_tmp:nn #1 }
```

13.18.1 Functions for setting the fake itemindent

__enumext_fake_item_indent:
__enumext_keyans_fake_item_indent:
__enumext_fake_item_indent_vii:
__enumext_fake_item_indent_viii:

The itemindent key does not set the value of \itemindent, it only sets the value of the *horizontal space* applied using \skip_horizontal:N. We will store this value in the variable and only apply it when it is greater than 0pt. Here I will need to place \mode_leave_vertical: and the plain TeX macro \ignorespaces to avoid unwanted extra space when using the itemindent key.

```
1056  \cs_set_protected:Nn \__enumext_fake_item_indent:
1057    {
1058      \dim_compare:nNnT
1059        { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
1060        >
1061        { \c_zero_dim }
1062        {
1063          \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
1064            {
1065              \exp_not:N \mode_leave_vertical:
1066              \exp_not:n { \skip_horizontal:n }
1067                { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
1068              \exp_not:N \ignorespaces
1069            }
1070        }
1071    }
1072  \cs_set_protected:Nn \__enumext_keyans_fake_item_indent:
1073    {
1074      \dim_compare:nNnT
1075        { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
1076        {
1077          \tl_set:Ne \l__enumext_fake_item_indent_v_tl
1078            {
1079              \exp_not:N \mode_leave_vertical:
1080              \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
```

```
1081             \exp_not:N \ignorespaces
1082           }
1083         }
1084     }
1085 \cs_set_protected:Nn \__enumext_fake_item_indent_vii:
1086   {
1087     \dim_compare:nNnT
1088       { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
1089       {
1090         \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
1091           {
1092             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
1093             \exp_not:N \ignorespaces
1094           }
1095       }
1096   }
1097 \cs_set_protected:Nn \__enumext_fake_item_indent_viii:
1098   {
1099     \dim_compare:nNnT
1100       { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
1101       {
1102         \tl_set:Ne \l__enumext_fake_item_indent_viii_tl
1103           {
1104             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
1105             \exp_not:N \ignorespaces
1106           }
1107       }
1108   }
```

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

13.19 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.

```
1109 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1110   {
1111     \keys_define:nn { enumext / #1 }
1112       {
1113         show-length .bool_set:c = { l__enumext_show_length_#2_bool },
1114         show-length .initial:n  = false,
1115       }
1116   }
1117 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

13.20 Setting before, after and first keys

before   Define and set before, before*, after and first keys for enumext, enumext*, keyans and keyans*
before*   environments.
after
first
```
1118 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1119   {
1120     \keys_define:nn { enumext / #1 }
1121       {
1122         before  .tl_set:c   = { l__enumext_before_no_starred_key_#2_tl },
1123         before  .value_required:n = true,
1124         before* .tl_set:c   = { l__enumext_before_starred_key_#2_tl },
1125         before* .value_required:n = true,
1126         after   .tl_set:c   = { l__enumext_after_stop_list_#2_tl },
1127         after   .value_required:n = true,
1128         first   .tl_set:c   = { l__enumext_after_list_args_#2_tl },
1129         first   .value_required:n = true,
1130       }
1131   }
1132 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

13.20.1   Functions for `before`, `after` and `first` keys in `enumext`

`\__enumext_before_args_exec:`
`\__enumext_before_keys_exec:`
`\__enumext_after_stop_list:`
`\__enumext_after_args_exec:`

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

```
1133 \cs_new_protected:Nn \__enumext_before_args_exec:
1134   {
1135     \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
1136   }
```

The function `\__enumext_before_keys_exec:` executes the {⟨*code*⟩} set by the `before` key *"before"* the `enumext` environment is started in *second argument* of the list. The {⟨*code*⟩} is executed *"knowing"* all definition and values provides by ⟨*keys*⟩: `\list`{⟨*arg one*⟩}{⟨*arg two*⟩{⟨*code*⟩}}

```
1137 \cs_new_protected:Nn \__enumext_before_keys_exec:
1138   {
1139     \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
1140   }
```

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

```
1141 \cs_new_protected:Nn \__enumext_after_stop_list:
1142   {
1143     \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
1144   }
```

The function `\__enumext_after_args_exec:` executes the {⟨*code*⟩} set by the `first` key after the end of the second argument of the list defining the `enumext` environment, just before the first occurrence of `\item`: `\list`{⟨*arg one*⟩}{⟨*arg two*⟩}{⟨*code*⟩}`\item`.

```
1145 \cs_new_protected:Nn \__enumext_after_args_exec:
1146   {
1147     \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
1148   }
```

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

13.20.2   Functions for `before`, `after` and `first` keys in `keyans`

`\__enumext_before_args_exec_v:`
`\__enumext_before_keys_exec_v:`
`\__enumext_after_stop_list_v:`
`\__enumext_after_args_exec_v:`

Same implementation as the one used in the `enumext` environment.

```
1149 \cs_new_protected:Nn \__enumext_before_args_exec_v:
1150   {
1151     \tl_use:N \l__enumext_before_starred_key_v_tl
1152   }
1153 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
1154   {
1155     \tl_use:N \l__enumext_before_no_starred_key_v_tl
1156   }
1157 \cs_new_protected:Nn \__enumext_after_stop_list_v:
1158   {
1159     \tl_use:N \l__enumext_after_stop_list_v_tl
1160   }
1161 \cs_new_protected:Nn \__enumext_after_args_exec_v:
1162   {
1163     \tl_use:N \l__enumext_after_list_args_v_tl
1164   }
```

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

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

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

Same implementation as the one used in the `enumext` environment.

```
1165 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
1166   {
1167     \tl_use:N \l__enumext_before_starred_key_vii_tl
1168   }
1169 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
1170   {
1171     \tl_use:N \l__enumext_before_starred_key_viii_tl
1172   }
1173 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
1174   {
1175     \tl_use:N \l__enumext_before_no_starred_key_vii_tl
1176   }
1177 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
1178   {
```

```
1179        \tl_use:N \l__enumext_before_no_starred_key_viii_tl
1180    }
1181 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
1182    {
1183        \tl_use:N \l__enumext_after_stop_list_vii_tl
1184    }
1185 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
1186    {
1187        \tl_use:N \l__enumext_after_stop_list_viii_tl
1188    }
1189 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
1190    {
1191        \tl_use:N \l__enumext_after_list_args_vii_tl
1192    }
1193 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
1194    {
1195        \tl_use:N \l__enumext_after_list_args_viii_tl
1196    }
```

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

13.21   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.

```
1197 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1198    {
1199        \keys_define:nn { enumext / #1 }
1200          {
1201            mini-env    .dim_set:c  = { l__enumext_minipage_right_#2_dim },
1202            mini-env    .value_required:n = true,
1203            mini-sep    .dim_set:c  = { l__enumext_minipage_hsep_#2_dim },
1204            mini-sep    .initial:n  = 0.3333em,
1205            mini-sep    .value_required:n = true,
1206            columns-sep .dim_set:c  = { l__enumext_columns_sep_#2_dim },
1207            columns-sep .value_required:n = true,
1208            columns     .int_set:c  = { l__enumext_columns_#2_int },
1209            columns     .initial:n  = 1,
1210            columns     .value_required:n = true,
1211          }
1212    }
1213 \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.

```
1214 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1215    {
1216        \keys_define:nn { enumext / #1 }
1217          {
1218            mini-right  .tl_gset:c = { g__enumext_miniright_code_#2_tl },
1219            mini-right  .value_required:n = true,
1220            mini-right* .code:n    = {
1221                                     \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
1222                                     \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1223                                    },
1224            mini-right* .value_required:n = true,
1225          }
1226    }
1227 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }
```

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

13.22   Adjustment of vertical spaces for `multicols`

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

To keep the desired spaces *above* and *below* in the *"list environment"* (`\topsep` + [`\partopsep`]) it is necessary to *"adjust"* the spaces added by the `multicols` environment. The most appropriate option in this case is to use a *"context sensitive"* vertical space with `\addvspace`.
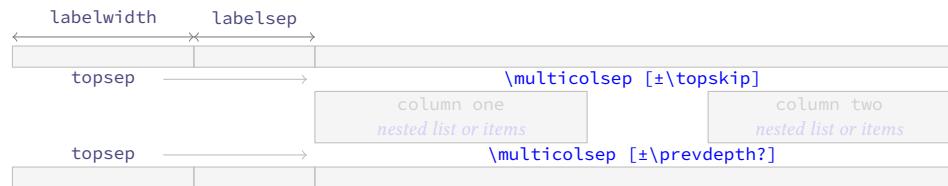
Figure 7: Representation of the vertical space in `multicols` for a nested level.

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

13.22.1 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*.

```
1228 \cs_new_protected:Nn \__enumext_multi_set_vskip:
1229   {
1230     \skip_set:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1231       {
1232         \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1233       }
1234     \skip_set:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1235       {
1236         \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1237       }
1238     \__enumext_add_pre_parsep:
1239   }
```

(*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*.

```
1240 \cs_new_protected:Nn \__enumext_add_pre_parsep:
1241   {
1242     \int_case:nn { \l__enumext_level_int }
1243       {
1244         { 2 }{
1245             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1246               {
1247                 \skip_add:Nn \l__enumext_multicols_above_ii_skip
1248                   {
1249                     \l__enumext_parsep_i_skip
1250                   }
1251               }
1252           }
1253         { 3 }{
1254             \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1255               {
1256                 \skip_add:Nn \l__enumext_multicols_above_iii_skip
1257                   {
1258                     \l__enumext_parsep_ii_skip
1259                   }
1260               }
1261           }
1262         { 4 }{
1263             \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1264               {
1265                 \skip_add:Nn \l__enumext_multicols_above_iv_skip
1266                   {
1267                     \l__enumext_parsep_iii_skip
1268                   }
```

```
1269                        }
1270                    }
1271                }
1272            }
```

(*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*⟩.

```
1273 \cs_new_protected:Nn \__enumext_multi_addvspace:
1274    {
1275      \__enumext_multi_set_vskip:
1276      \mode_if_vertical:T
1277        {
1278          \skip_add:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1279            {
1280              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1281            }
1282          \skip_add:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1283            {
1284              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1285            }
1286        }
1287      \par\nopagebreak
1288      \addvspace{ \skip_use:c { l__enumext_multicols_above_ \__enumext_level: _skip } }
1289    }
```

(*End of definition for* __enumext_multi_addvspace:.)

13.22.2 Adjustment of vertical spaces for multicols in keyans

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

```
1290 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1291    {
1292      \skip_set:Nn \l__enumext_multicols_above_v_skip
1293        {
1294          \l__enumext_topsep_v_skip
1295        }
1296      \skip_set:Nn \l__enumext_multicols_below_v_skip
1297        {
1298          \l__enumext_topsep_v_skip
1299        }
1300    }
1301 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1302    {
1303      \__enumext_keyans_multi_set_vskip:
1304      \mode_if_vertical:T
1305        {
1306          \skip_add:Nn \l__enumext_multicols_above_v_skip
1307            {
1308              \skip_use:N \l__enumext_partopsep_v_skip
1309            }
1310          \skip_add:Nn \l__enumext_multicols_below_v_skip
1311            {
1312              \skip_use:N \l__enumext_partopsep_v_skip
1313            }
1314        }
1315      \par\nopagebreak
1316      \addvspace{ \l__enumext_multicols_above_v_skip }
1317    }
```

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

13.23 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.

13.23.1   Adjustment of vertical spaces for `minipage` in enumext

`\__enumext_minipage_set_skip:`
`\__enumext_minipage_add_space:`

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

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

```
1318 \cs_new_protected:Nn \__enumext_minipage_set_skip:
1319   {
1320     \skip_set:Nn \l__enumext_minipage_right_skip
1321       {
1322         \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1323       }
1324     \mode_if_vertical:T
1325       {
1326         \skip_add:Nn \l__enumext_minipage_right_skip
1327           {
1328             \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1329           }
1330       }
1331     \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip
```

We will adjust the values `\l__enumext_multicols_above_X_skip` and `\l__enumext_multicols_-below_X_skip` and call the function `\__enumext_pre_itemsep_skip:`.

```
1332     \skip_set_eq:cN
1333       { l__enumext_multicols_above_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1334     \skip_set_eq:cN
1335       { l__enumext_multicols_below_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1336     \__enumext_pre_itemsep_skip:
```

If the environment `multicols` is active, we set `\topskip=0pt` and then we make `\multicolsep` have the same value as `\l__enumext_multicols_above_X_skip`.

```
1337     \int_compare:nNnT
1338       { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1339       {
1340         \skip_zero:N \topskip
1341         \skip_set_eq:Nc \multicolsep { l__enumext_multicols_above_ \__enumext_level: _skip }
1342       }
1343   }
```

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

```
1344 \cs_new_protected:Nn \__enumext_minipage_add_space:
1345   {
1346     \__enumext_minipage_set_skip:
1347     \__enumext_unskip_unkern:
1348     \mode_if_vertical:TF
1349       {
1350         \nopagebreak\nointerlineskip
```

```
1351          }
1352        {
1353            \par\nopagebreak\nointerlineskip
1354            \skip_zero:c { l__enumext_partopsep_ \__enumext_level: _skip }
1355        }
1356      \int_compare:nNnTF
1357        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1358        {
1359            \addvspace{ 0.445\box_ht:N \strutbox }
1360        }
1361        {
1362            \addvspace{ 0.250\box_ht:N \strutbox }
1363        }
1364    }
```

(*End of definition for* __enumext_minipage_set_skip: *and* __enumext_minipage_add_space:*.*)

__enumext_pre_itemsep_skip: The function __enumext_pre_itemsep_skip: will adjust the spaces below the environment minipage and the environment multicols if it is nested in it, taking into account the value of \itemsep from the previous level.

```
1365  \cs_new_protected:Nn \__enumext_pre_itemsep_skip:
1366    {
1367      \int_case:nn { \l__enumext_level_int }
1368        {
1369          { 2 }{
1370                \skip_if_eq:nnTF
1371                  { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1372                  {
1373                    \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1374                    \skip_set:Nn \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1375                  }
1376                  {
1377                    \dim_compare:nNnT
1378                      { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1379                      {
1380                        \skip_sub:Nn
1381                          \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1382                        \skip_sub:Nn
1383                          \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1384                        \skip_add:Nn
1385                          \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1386                        \skip_add:Nn
1387                          \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1388                      }
1389                    \dim_compare:nNnT
1390                      { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1391                      {
1392                        \skip_set:Nn \l__enumext_minipage_temp_skip
1393                          {
1394                            \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1395                          }
1396                        \skip_sub:Nn
1397                          \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1398                        \skip_sub:Nn
1399                          \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1400                        \skip_add:Nn
1401                          \l__enumext_minipage_after_skip
1402                          { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1403                        \skip_add:Nn
1404                          \l__enumext_multicols_below_ii_skip
1405                          { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1406                      }
1407                  }
1408              }
1409          { 3 }{
1410                \skip_if_eq:nnTF
1411                  { \l__enumext_itemsep_ii_skip } { \c_zero_skip }
1412                  {
1413                    \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1414                    \skip_set:Nn \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1415                  }
```

```
1416                            {
1417                                \dim_compare:nNnT
1418                                  { \l__enumext_itemsep_ii_skip } < { \l__enumext_minipage_after_skip }
1419                                  {
1420                                    \skip_sub:Nn
1421                                      \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1422                                    \skip_sub:Nn
1423                                      \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1424                                    \skip_add:Nn
1425                                      \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1426                                    \skip_add:Nn
1427                                      \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1428                                  }
1429                                \dim_compare:nNnT
1430                                  { \l__enumext_itemsep_ii_skip } > { \l__enumext_minipage_after_skip }
1431                                  {
1432                                    \skip_set:Nn \l__enumext_minipage_temp_skip
1433                                      {
1434                                        \l__enumext_itemsep_ii_skip - \l__enumext_minipage_after_skip
1435                                      }
1436                                    \skip_sub:Nn
1437                                      \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1438                                    \skip_sub:Nn
1439                                      \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1440                                    \skip_add:Nn
1441                                      \l__enumext_minipage_after_skip
1442                                      { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1443                                    \skip_add:Nn
1444                                      \l__enumext_multicols_below_iii_skip
1445                                      { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1446                                  }
1447                              }
1448                          }
1449                { 4 }{
1450                        \skip_if_eq:nnTF { \l__enumext_itemsep_iii_skip } { \c_zero_skip }
1451                          {
1452                            \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1453                            \skip_set:Nn \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1454                          }
1455                          {
1456                            \dim_compare:nNnT
1457                              { \l__enumext_itemsep_iii_skip } < { \l__enumext_minipage_after_skip }
1458                              {
1459                                \skip_sub:Nn
1460                                  \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1461                                \skip_sub:Nn
1462                                  \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1463                                \skip_add:Nn
1464                                  \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1465                                \skip_add:Nn
1466                                  \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1467                              }
1468                            \dim_compare:nNnT
1469                              { \l__enumext_itemsep_iii_skip } > { \l__enumext_minipage_after_skip }
1470                              {
1471                                \skip_set:Nn \l__enumext_minipage_temp_skip
1472                                  {
1473                                    \l__enumext_itemsep_iii_skip - \l__enumext_minipage_after_skip
1474                                  }
1475                                \skip_sub:Nn
1476                                  \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1477                                \skip_sub:Nn
1478                                  \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1479                                \skip_add:Nn
1480                                  \l__enumext_minipage_after_skip
1481                                  { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1482                                \skip_add:Nn
1483                                  \l__enumext_multicols_below_iv_skip
1484                                  { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1485                              }
1486                          }
```

```
1487                        }
1488                }
1489        }
```

(*End of definition for* __enumext_pre_itemsep_skip:.)

13.23.2 Adjustment of vertical spaces for `minipage` in keyans

__enumext_keyans_minipage_set_skip:
__enumext_keyans_minipage_add_space:
__enumext_keyans_pre_itemsep_skip:

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

```
1490  \cs_new_protected:Nn \__enumext_keyans_minipage_set_skip:
1491    {
1492      \skip_zero:N \l__enumext_minipage_after_skip
1493      \skip_zero:N \l__enumext_minipage_left_skip
1494      \skip_zero:N \l__enumext_minipage_right_skip
1495      \skip_set:Nn \l__enumext_minipage_right_skip
1496        {
1497          \l__enumext_topsep_v_skip
1498        }
1499      \mode_if_vertical:T
1500        {
1501          \skip_add:Nn \l__enumext_minipage_right_skip
1502            {
1503              \l__enumext_partopsep_v_skip
1504            }
1505        }
1506      \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip
1507      \skip_set_eq:NN \l__enumext_multicols_above_v_skip \l__enumext_minipage_right_skip
1508      \skip_set_eq:NN \l__enumext_multicols_below_v_skip \l__enumext_minipage_right_skip
1509      \__enumext_keyans_pre_itemsep_skip:
1510      \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
1511        {
1512          \skip_zero:N \topskip
1513          \skip_set_eq:NN \multicolsep \l__enumext_minipage_right_skip
1514        }
1515    }
1516  \cs_new_protected:Nn \__enumext_keyans_minipage_add_space:
1517    {
1518      \__enumext_keyans_minipage_set_skip:
1519      \__enumext_unskip_unkern:
1520      \mode_if_vertical:TF
1521        {
1522          \nopagebreak\nointerlineskip
1523        }
1524        {
1525          \par\nopagebreak\nointerlineskip
1526          \skip_zero:N \l__enumext_partopsep_v_skip
1527        }
1528      \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1529        {
1530          \addvspace{ 0.445\box_ht:N \strutbox }
1531        }
1532        {
1533          \addvspace{ 0.250\box_ht:N \strutbox }
1534        }
1535    }
1536  \cs_new_protected:Nn \__enumext_keyans_pre_itemsep_skip:
1537    {
1538      \skip_if_eq:nnTF
1539        { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1540        {
1541          \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1542          \skip_set:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1543        }
1544        {
1545          \dim_compare:nNnT
1546            { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1547            {
1548              \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1549              \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1550              \skip_add:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
```

```
1551                    \skip_add:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1552                  }
1553              \dim_compare:nNnT
1554                { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1555                {
1556                  \skip_set:Nn \l__enumext_minipage_temp_skip
1557                    {
1558                      \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1559                    }
1560                  \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1561                  \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1562                  \skip_add:Nn \l__enumext_minipage_after_skip
1563                    { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1564                  \skip_add:Nn \l__enumext_multicols_below_v_skip
1565                    { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1566                }
1567            }
1568        }
```

(*End of definition for* __enumext_keyans_minipage_set_skip:, __enumext_keyans_minipage_add_space:, *and* __enu-
mext_keyans_pre_itemsep_skip:.)

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

```
1569  \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1570    {
1571      \skip_zero_new:N \l__enumext_minipage_left_skip
1572      \skip_gzero_new:N \g__enumext_minipage_right_skip
1573      \skip_gzero_new:N \g__enumext_minipage_after_skip
1574      \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1575        {
1576          \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1577          \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1578        }
1579        {
1580          \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1581          \skip_gset:Nn \g__enumext_minipage_right_skip
1582            {
1583              \l__enumext_topsep_vii_skip
1584            }
1585          \skip_gset:Nn \g__enumext_minipage_after_skip
1586            {
1587              0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1588            }
1589        }
1590    }
1591  \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1592    {
1593      \skip_zero_new:N \l__enumext_minipage_after_skip
1594      \skip_zero_new:N \l__enumext_minipage_left_skip
1595      \skip_zero_new:N \l__enumext_minipage_right_skip
1596      \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1597        {
1598          \skip_set:Nn \l__enumext_minipage_left_skip
1599            {
1600              0.5\box_dp:N \strutbox
1601            }
1602          \skip_set:Nn \l__enumext_minipage_right_skip
1603            {
1604              \l__enumext_partopsep_viii_skip
1605            }
1606          \skip_set:Nn \l__enumext_minipage_after_skip
1607            {
1608              1.6\box_dp:N \strutbox
1609            }
1610        }
1611        {
1612          \skip_set:Nn \l__enumext_minipage_left_skip
1613            {
```

```
1614                    0.5875\box_dp:N \strutbox
1615                 }
1616            \skip_set:Nn \l__enumext_minipage_right_skip
1617               {
1618                 \l__enumext_topsep_viii_skip
1619               }
1620            \skip_set:Nn \l__enumext_minipage_after_skip
1621               {
1622                 0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1623               }
1624          }
1625       }
```

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

__enumext_mini_addvspace_vii:
__enumext_mini_addvspace_viii:

The functions __enumext_mini_addvspace_vii: and __enumext_mini_addvspace_viii: will apply the vertical space *"only above"* the __enumext_mini_page environment on the *left side* when the mini-right key is active in the enumext* and keyans* environments.

Here we will NOT take into account whether TeX is in ⟨*horizontal mode*⟩ or ⟨*vertical mode*⟩, since \partopsep is equal to 0pt in both environments.

```
1626  \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1627     {
1628        \__enumext_mini_set_vskip_vii:
1629        \par\nopagebreak
1630        \addvspace { \l__enumext_minipage_left_skip }
1631     }
1632  \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1633     {
1634        \__enumext_mini_set_vskip_viii:
1635        \par\nopagebreak
1636        \addvspace { \l__enumext_minipage_left_skip }
1637     }
```

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

13.23.4 The command \miniright

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

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

```
1638  \NewDocumentCommand \miniright { s }
1639     {
1640        \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1641           {
1642              \msg_error:nnn { enumext } { wrong-miniright-place }
1643           }
1644        % outside
1645        \bool_lazy_and:nnT
1646           { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
1647           { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
1648           {
1649              \msg_error:nnn { enumext } { wrong-miniright-place }
1650           }
1651        % starred env
1652        \bool_lazy_and:nnT
1653           { \bool_if_p:N \g__enumext_starred_bool }
1654           { \bool_not_p:n { \l__enumext_standar_bool } }
1655           {
1656              \msg_error:nnn { enumext } { wrong-miniright-starred }
1657           }
1658        % exec
1659        \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
1660           {
1661              \__enumext_keyans_mini_right_cmd:n {#1}
1662           }
```

```
1663      { \__enumext_mini_right_cmd:n {#1} }
1664    }
```

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

__enumext_mini_right_cmd:n   The function __enumext_mini_right_cmd:n takes as argument the *starred* '*' of the \miniright command in the enumext environment. We check if the mini-env key is active via the variable \l__enumext_-minipage_right_X_dim, if so we close the multicols environment with the __enumext_mini_page environment on the *"left side"*, then we open the __enumext_mini_page environment on the *"right side"*, apply our adjusted *"vertical spaces"*, followed by adding the \centering command when the *starred argument* '*' is not present and set zero \g__enumext_minipage_stat_int, otherwise we return an error.

```
1665 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1666    {
1667      \dim_compare:nNnTF
1668        { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1669        {
1670          \__enumext_multicols_stop:
1671          \int_compare:nNnT
1672            { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } = { 1 }
1673            {
1674              \par\addvspace{ \l__enumext_minipage_after_skip }
1675            }
1676          \end__enumext_mini_page
1677          \hfill
1678          \__enumext_mini_page{ \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } }
1679            \par\nointerlineskip
1680            \addvspace { \l__enumext_minipage_right_skip }
1681            \bool_if:nF {#1}
1682              {
1683                \centering
1684              }
1685            \int_gzero:N \g__enumext_minipage_stat_int
1686        }
1687        { \msg_error:nnn { enumext } { wrong-miniright-use } }
1688      % paranoia
1689      \RenewDocumentCommand \miniright { s }
1690        {
1691          \msg_error:nn { enumext } { many-miniright-used }
1692        }
1693    }
```

(*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.

```
1694 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1695    {
1696      \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1697        {
1698          \__enumext_keyans_multicols_stop:
1699          \int_compare:nNnT { \l__enumext_columns_v_int } = { 1 }
1700            {
1701              \par\addvspace{ \l__enumext_minipage_after_skip }
1702            }
1703          \end__enumext_mini_page
1704          \hfill
1705          \__enumext_mini_page{ \l__enumext_minipage_right_v_dim }
1706            \par\nointerlineskip
1707            \addvspace { \l__enumext_minipage_right_skip }
1708            \bool_if:nF {#1}
1709              {
1710                \centering
1711              }
1712            \int_gzero:N \g__enumext_minipage_stat_int
1713        }
1714        { \msg_error:nnn { enumext } { wrong-miniright-use } }
1715      % paranoia
1716      \RenewDocumentCommand \miniright { s }
1717        {
1718          \msg_error:nn { enumext } { many-miniright-used }
```

```
1719        }
1720     }
```

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

13.24   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.

```
1721 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1722   {
1723     \keys_define:nn { enumext / #1 }
1724       {
1725         above  .skip_set:c = { l__enumext_vspace_above_#2_skip },
1726         above  .value_required:n = true,
1727         above* .code:n     = \bool_set_true:c { l__enumext_vspace_a_star_#2_bool }
1728                             \keys_set:nn { enumext / #1 } { above = {##1} },
1729         above* .value_required:n = true,
1730         below  .skip_set:c = { l__enumext_vspace_below_#2_skip },
1731         below  .value_required:n = true,
1732         below* .code:n     = \bool_set_true:c { l__enumext_vspace_b_star_#2_bool }
1733                             \keys_set:nn { enumext / #1 } { below = {##1} },
1734         below* .value_required:n = true,
1735       }
1736   }
1737 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

13.24.1   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.

```
1738 \cs_new_protected:Nn \__enumext_vspace_above:
1739   {
1740     \skip_if_eq:nnF
1741       { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1742       {
1743         \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
1744           {
1745             \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1746           }
1747           {
1748             \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1749           }
1750       }
1751   }
```

(*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.

```
1752 \cs_new_protected:Nn \__enumext_vspace_below:
1753   {
1754     \skip_if_eq:nnF
1755       { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1756       {
1757         \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1758           {
1759             \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1760           }
1761           {
1762             \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1763           }
1764       }
1765   }
```

(*End of definition for* __enumext_vspace_below:.)

13.24.2 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.

```
1766 \cs_new_protected:Nn \__enumext_vspace_above_v:
1767   {
1768     \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1769       {
1770         \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1771           {
1772             \vspace*{ \l__enumext_vspace_above_v_skip }
1773           }
1774           { \vspace { \l__enumext_vspace_above_v_skip } }
1775       }
1776   }
```

(*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.

```
1777 \cs_new_protected:Nn \__enumext_vspace_below_v:
1778   {
1779     \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1780       {
1781         \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1782           {
1783             \vspace*{ \l__enumext_vspace_below_v_skip }
1784           }
1785           { \vspace { \l__enumext_vspace_below_v_skip } }
1786       }
1787   }
```

(*End of definition for* __enumext_vspace_below_v:.)

13.24.3 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.

```
1788 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1789   {
1790     \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1791       {
1792         \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1793           {
1794             \vspace*{ \l__enumext_vspace_above_vii_skip }
1795           }
1796           { \vspace { \l__enumext_vspace_above_vii_skip } }
1797       }
1798   }
1799 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1800   {
1801     \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1802       {
1803         \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1804           {
1805             \vspace*{ \l__enumext_vspace_above_viii_skip }
1806           }
1807           { \vspace { \l__enumext_vspace_above_viii_skip } }
1808       }
1809   }
```

(*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.

```
1810 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1811   {
1812     \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1813       {
1814         \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1815           {
1816             \vspace*{ \l__enumext_vspace_below_vii_skip }
```

```
1817                }
1818              { \vspace { \l__enumext_vspace_below_vii_skip } }
1819          }
1820      }
1821  \cs_new_protected:Nn \__enumext_vspace_below_viii:
1822      {
1823        \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1824          {
1825            \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1826              {
1827                \vspace*{ \l__enumext_vspace_below_viii_skip }
1828              }
1829              { \vspace { \l__enumext_vspace_below_viii_skip } }
1830          }
1831      }
```

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

13.25   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 We define the keys `series`, `resume` and `resume*` only for the *"first level"* of `enumext` and `enumext*`.
resume
resume*
```
1832  \cs_set_protected:Npn \__enumext_tmp:n #1
1833      {
1834        \keys_define:nn { enumext / #1 }
1835          {
1836            series  .str_set:N = \l__enumext_series_str,
1837            series  .value_required:n = true,
1838            resume  .code:n = \__enumext_resume_series:n {##1},
1839            resume* .code:n = \__enumext_resume_starred:,
1840            resume* .value_forbidden:n = true,
1841          }
1842      }
1843  \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

13.25.1   Internal functions for series key

__enumext_filter_series:n   The function `\__enumext_filter_series:n` will be in charge of filtering the ⟨*keys*⟩ we want to store where
__enumext_filter_series_key:n   `{#1}` represents the *optional argument* passed to the environment.
__enumext_filter_series_pair:nn
```
1844  \cs_new:Npn \__enumext_filter_series:n #1
1845      {
1846        \use:e
1847          {
1848            \keyval_parse:NNn
1849              \__enumext_filter_series_key:n
1850              \__enumext_filter_series_pair:nn {#1}
1851          }
1852      }
```

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.

```
1853  \cs_new:Npn \__enumext_filter_series_key:n #1
1854      {
1855        \str_case:nnF {#1}
1856          {
1857            { resume } {} { resume* } {} { base-fix } {}
1858          }
1859          { , { \exp_not:n {#1} } }
1860      }
```

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

```
1861  \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1862      {
1863        \str_case:nnF {#1}
1864          {
1865            { series } {} { resume } {} { start } {}
```

```
1866          { start* } {}   { save-ans } {} { save-key } {}
1867          }
1868          { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1869      }
```

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

__enumext_parse_series:n
__enumext_resume_last:n

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

💣 The value of boolean variable \l__enumext_resume_active_bool is set to true by the function __enumext_resume_-counter:n which is used by the keys resume and resume*, in this case we must Make sure it is set to false so that it does not overwrite the default filtered ⟨keys⟩. This function is passed to the function __enumext_parse_keys:n in the enumext environment definition (§13.39) and to the function __enumext_parse_keys_vii:n in the enumext* environment definition (§13.44).

```
1870 \cs_new_protected:Npn \__enumext_parse_series:n #1
1871   {
1872     \str_if_empty:NTF \l__enumext_series_str
1873       {
1874         \bool_if:NF \l__enumext_resume_active_bool
1875           {
1876             \__enumext_resume_last:n {#1}
1877           }
1878       }
1879       {
1880         \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str _tl }
1881         \tl_gset:ce { g__enumext_series_ \l__enumext_series_str _tl }
1882           { \__enumext_filter_series:n {#1} }
1883         \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str _int }
1884           {
1885             \int_new:c { g__enumext_series_ \l__enumext_series_str _int }
1886           }
1887       }
1888   }
```

The function __enumext_resume_last:n will be in charge of saving the filtering ⟨keys⟩ when the series key is *not used* and will save them in the variable \g__enumext_standar_series_tl for the enumext environment and in the variable \g__enumext_starred_series_tl for the enumext* environment.

```
1889 \cs_new_protected:Npn \__enumext_resume_last:n #1
1890   {
1891     \bool_if:NT \l__enumext_standar_first_bool
1892       {
1893         \tl_gclear:N \g__enumext_standar_series_tl
1894         \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1895       }
1896     \bool_if:NT \l__enumext_starred_first_bool
1897       {
1898         \tl_gclear:N \g__enumext_starred_series_tl
1899         \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1900       }
1901   }
```

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

13.25.2 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 (§13.39) and the enumext* environment definition (§13.44).

```
1902 \cs_new_protected:Nn \__enumext_resume_save_counter:
1903   {
```

```
1904     \bool_if:NT \g__enumext_standar_bool
1905       {
1906         \tl_if_empty:NF \l__enumext_series_str
1907           {
1908             \int_gset_eq:cN
1909               { g__enumext_series_ \l__enumext_series_str _int } \value{enumXi}
1910           }
1911         \tl_if_empty:NTF \l__enumext_resume_name_tl
1912           {
1913             \str_if_empty:NT \l__enumext_series_str
1914               {
1915                 \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1916               }
1917           }
1918           {
1919             \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1920               {
1921                 \int_gset_eq:cN
1922                   { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXi}
1923               }
1924           }
1925         \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1926           {
1927             \int_gset_eq:cN
1928               { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1929           }
1930       }
1931     \bool_if:NT \g__enumext_starred_bool
1932       {
1933         \tl_if_empty:NF \l__enumext_series_str
1934           {
1935             \int_gset_eq:cN
1936               { g__enumext_series_ \l__enumext_series_str _int } \value{enumXvii}
1937           }
1938         \tl_if_empty:NTF \l__enumext_resume_name_tl
1939           {
1940             \str_if_empty:NT \l__enumext_series_str
1941               {
1942                 \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1943               }
1944           }
1945           {
1946             \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1947               {
1948                 \int_gset_eq:cN
1949                   { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXvii}
1950               }
1951           }
1952         \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1953           {
1954             \int_gset_eq:cN
1955               { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1956           }
1957       }
1958   }
```

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

13.25.3 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.

```
1959   \cs_new_protected:Npn \__enumext_resume_series:n #1
1960     {
1961       \tl_if_empty:nTF {#1}
1962         {
1963           \__enumext_resume_counter:n { }
```

```
1964        }
1965        {
1966          \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1967            {
1968              \__enumext_resume_counter:n {#1}
1969              \bool_if:NT \g__enumext_standar_bool
1970                {
1971                  \keys_set:nv { enumext / level-1 }
1972                    { g__enumext_series_ \tl_to_str:n {#1} _tl }
1973                }
1974              \bool_if:NT \g__enumext_starred_bool
1975                {
1976                  \keys_set:nv { enumext / enumext* }
1977                    { g__enumext_series_ \tl_to_str:n {#1} _tl }
1978                }
1979            }
1980            {
1981              \bool_if:NT \g__enumext_standar_bool
1982                {
1983                  \msg_error:nnn { enumext } { unknown-series } {#1}
1984                }
1985              \bool_if:NT \g__enumext_starred_bool
1986                {
1987                  \msg_error:nnn { enumext } { unknown-series } {#1}
1988                }
1989            }
1990        }
1991    }
```

(*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.

```
1992 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1993    {
1994      \bool_set_true:N \l__enumext_resume_active_bool
1995      \tl_set:Nn \l__enumext_resume_name_tl {#1}
1996      \tl_if_empty:NTF \l__enumext_resume_name_tl
1997        {
1998          \__enumext_resume_counter:
1999        }
2000        {
2001          \__enumext_resume_counter_series:
2002        }
2003      \__enumext_resume_counter_save_ans:
2004    }
```

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.

```
2005 \cs_new_protected:Nn \__enumext_resume_counter:
2006    {
2007      \bool_if:NT \g__enumext_standar_bool
2008        {
2009          \int_gincr:N \g__enumext_resume_int
2010          \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
2011        }
2012      \bool_if:NT \g__enumext_starred_bool
2013        {
2014          \int_gincr:N \g__enumext_resume_vii_int
2015          \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
2016        }
2017    }
```

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.

```
2018 \cs_new_protected:Nn \__enumext_resume_counter_series:
2019   {
2020     \bool_if:NT \g__enumext_standar_bool
2021       {
2022         \int_set:Nn \l__enumext_start_i_int
2023           {
2024             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
2025           }
2026       }
2027     \bool_if:NT \g__enumext_starred_bool
2028       {
2029         \int_set:Nn \l__enumext_start_vii_int
2030           {
2031             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
2032           }
2033       }
2034   }
```

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.

```
2035 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
2036   {
2037     \bool_lazy_and:nnT
2038       { \bool_if_p:N \l__enumext_standar_first_bool }
2039       { \bool_if_p:N \l__enumext_store_active_bool }
2040       {
2041         \int_set:Nn \l__enumext_start_i_int
2042           {
2043             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
2044           }
2045       }
2046     \bool_lazy_and:nnT
2047       { \bool_if_p:N \l__enumext_starred_first_bool }
2048       { \bool_if_p:N \l__enumext_store_active_bool }
2049       {
2050         \int_set:Nn \l__enumext_start_vii_int
2051           {
2052             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
2053           }
2054       }
2055   }
```

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

13.25.4 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.

```
2056 \cs_new_protected:Nn \__enumext_resume_starred:
2057   {
2058     \bool_if:NT \g__enumext_standar_bool
2059       {
2060         \tl_if_empty:NF \g__enumext_standar_series_tl
2061           {
2062             \__enumext_resume_counter:n { }
2063             \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
2064           }
2065       }
2066     \bool_if:NT \g__enumext_starred_bool
2067       {
2068         \tl_if_empty:NF \g__enumext_starred_series_tl
2069           {
2070             \__enumext_resume_counter:n { }
2071             \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
2072           }
2073       }
2074   }
```

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

13.26 Setting save-ans, check-ans and no-store keys

The key save-ans is directly associated with the keys check-ans, no-store, resume and resume*, this will activate the entire *"storage system"* in the enumext package.

13.26.1 Setting save-ans key

save-ans

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

```
2075 \cs_set_protected:Npn \__enumext_tmp:n #1
2076   {
2077     \keys_define:nn { enumext / #1 }
2078       {
2079         save-ans .code:n = \__enumext_storing_set:n {##1},
2080         save-ans .value_required:n = true,
2081       }
2082   }
2083 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

13.26.2 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:.

```
2084 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
2085   {
2086     \msg_term:nnVV { enumext } { save-ans-log }
2087       \g__enumext_envir_name_tl \l__enumext_store_name_tl
2088   }
2089 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
2090   {
2091     \msg_term:nnVV { enumext } { save-ans-log-hook }
2092       \g__enumext_envir_name_tl \g__enumext_store_name_tl
2093   }
```

(*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.

```
2094 \cs_new_protected:Npn \__enumext_storing_set:n #1
2095   {
2096     \tl_set:Ne \l__enumext_store_name_tl {#1}
2097     \tl_if_empty:NTF \l__enumext_store_name_tl
2098       {
2099         \bool_lazy_or:nnT
2100           { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
2101           {
2102             \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
2103           }
2104       }
2105       {
2106         \bool_lazy_or:nnT
2107           { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
2108           {
2109             \__enumext_start_save_ans_msg:
2110             \__enumext_storing_exec:
2111           }
2112       }
2113   }
```

The function __enumext_storing_exec: will set to true the variable \l__enumext_store_active_-bool which activates the use of the \anskey command and the anskey*, keyans, keyans* and keyanspic environments and will set to "true" the variable \l__enumext_check_answers_bool used for internal checking answers mechanism set by the check-ans and no-store keys, copy {⟨*store name*⟩} into the variable \g__enumext_store_name_tl and execute the function __enumext_anskey_env_make:V creating the environment anskey* (§13.31).

```
2114 \cs_new_protected:Nn \__enumext_storing_exec:
```

```
2115    {
2116      \bool_set_true:N \l__enumext_store_active_bool
2117      \bool_set_true:N \l__enumext_check_answers_bool
2118      \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
2119      \__enumext_anskey_env_make:V \l__enumext_store_name_tl
```

The *prop list* \g__enumext_series_⟨*store name*⟩_prop and the *sequence* \g__enumext_series_⟨*store name*⟩_seq will be created globally to *"store content"* in case they do not exist together with the integer variable \g__enumext_series_⟨*store name*⟩_int used by the keys resume and resume*.

```
2120      \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
2121        {
2122          \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
2123          \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
2124        }
2125      \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
2126        {
2127          \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
2128          \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
2129        }
2130      \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
2131        {
2132          \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
2133          \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
2134        }
2135    }
```

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

13.26.3   The check answer mechanism

The internal mechanism for *"checking answers"* follows this logic:

> If the line begins with \item or \item* and does NOT *open a nested environment*, each \item or \item* must contain a *single* execution of the \anskey command, i.e. the counter of the executions of the \anskey command must be equal to the counter associated with the sum of executions of \item and \item*.

> If the line begins with \item or \item* and *opens a nested environment* each \item or \item* in the nested environment must have a *single* execution of the \anskey command and the counter associated to the sum of \item and \item* executions must decrementing by *"one"* to maintain equality.

In order for the mechanism for the check-answer to work (not counting keyans, keyans* and keyanspic) we need:

1. We must keep track of the total number of \item and \item* (enumerated) that appear within the environment including the nested levels.
2. We must keep track of the total number of \item and \item* (enumerated) that appear per level of nesting.
3. Keeping track of the number of times the environment nests.

The integer variable associated to the sum of each \item and \item* in the environment \g__enumext_item_number_int must match the integer variable \g__enumext_item_anskey_int associated to the execution of the command \anskey. We analyze the cases:

a) If the list only has one level the number of \item + \item* = \anskey
b) If the list has *nested levels*, for each level of nesting we need to decrementing by one (for the \item or \item* that opens the nest) so that the account remains the same.

With keyans, keyans* and keyanspic it is enough to increase in one the integer of \anskey. The integers created must be global if they are not lost in the interior levels of nesting and to execute the test we will use a *"hook"* function after closing the *first level* of the environment.

13.26.4   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.

```
2136  \cs_set_protected:Npn \__enumext_tmp:n #1
2137    {
2138      \keys_define:nn { enumext / #1 }
2139        {
2140          check-ans .bool_set:N = \l__enumext_check_ans_key_bool,
2141          check-ans .initial:n  = false,
2142          check-ans .value_required:n = true,
2143          no-store  .code:n = {
2144                              \bool_set_false:N \l__enumext_check_answers_bool
```

```
2145                              \bool_set_false:N \l__enumext_check_ans_key_bool
2146                          },
2147          no-store  .value_forbidden:n = true,
2148      }
2149  }
2150  \clist_map_inline:nn
2151    {
2152      level-1, level-2, level-3, level-4, enumext*
2153    }
2154    { \__enumext_tmp:n {#1} }
```

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

13.26.5 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.

```
2155  \cs_new_protected:Nn \__enumext_check_ans_active:
2156    {
2157      \tl_if_empty:NF \l__enumext_store_name_tl
2158        {
2159          \bool_if:NT \l__enumext_check_answers_bool
2160            {
2161              \__enumext_check_ans_level:
2162            }
2163        }
2164    }
```

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"* .

```
2165  \cs_new_protected:Nn \__enumext_check_ans_level:
2166    {
2167      \int_case:nn { \l__enumext_level_int }
2168        {
2169          { 1 }{
2170              \bool_lazy_all:nT
2171                {
2172                  { \bool_if_p:N \g__enumext_starred_bool }
2173                  { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
2174                }
2175                {
2176                  \int_gdecr:N \g__enumext_item_number_int
2177                  \bool_set_false:N \l__enumext_item_number_bool
2178                }
2179            }
2180          { 2 }{
2181              \int_gdecr:N \g__enumext_item_number_int
2182              \bool_set_false:N \l__enumext_item_number_bool
2183            }
2184          { 3 }{
2185              \int_gdecr:N \g__enumext_item_number_int
2186              \bool_set_false:N \l__enumext_item_number_bool
2187            }
2188          { 4 }{
2189              \int_gdecr:N \g__enumext_item_number_int
2190              \bool_set_false:N \l__enumext_item_number_bool
2191            }
2192        }
```

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.

```
2193      \int_case:nn { \l__enumext_level_h_int }
2194        {
2195          { 1 }{
2196              \bool_lazy_all:nT
2197                {
2198                  { \bool_if_p:N \g__enumext_standar_bool }
2199                  { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
```

```
2200                        }
2201                        {
2202                          \int_gdecr:N \g__enumext_item_number_int
2203                          \bool_set_false:N \l__enumext_item_number_bool
2204                        }
2205                    }
2206                }
2207        }
```

(*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.

```
2208 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
2209    {
2210      \bool_lazy_and:nnT
2211        { \bool_if_p:N \l__enumext_check_ans_key_bool }
2212        { \bool_if_p:N \g__enumext_standar_bool }
2213        {
2214          \bool_gset_true:N \g__enumext_check_ans_key_bool
2215        }
2216      \bool_lazy_and:nnT
2217        { \bool_if_p:N \l__enumext_check_ans_key_bool }
2218        { \bool_if_p:N \g__enumext_starred_bool }
2219        {
2220          \bool_gset_true:N \g__enumext_check_ans_key_bool
2221        }
2222    }
```

(*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:.

```
2223 \cs_new_protected:Nn \__enumext_item_answer_diff:
2224    {
2225      \int_gset:Nn \g__enumext_item_answer_diff_int
2226        {
2227          \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
2228        }
2229    }
```

(*End of definition for* __enumext_item_answer_diff:.)

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

__enumext_check_ans_msg_less:
__enumext_check_ans_msg_same_ok:
__enumext_check_ans_msg_greater:

```
2230 \cs_new_protected:Nn \__enumext_check_ans_show:
2231    {
2232      \int_case:nn { \g__enumext_item_answer_diff_int }
2233        {
2234          { -1 }{ \__enumext_check_ans_msg_less:     }
2235          {  0 }{ \__enumext_check_ans_msg_same_ok: }
2236          {  1 }{ \__enumext_check_ans_msg_greater: }
2237        }
2238    }
2239 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
2240    {
2241      \msg_warning:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2242        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2243    }
2244 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
2245    {
2246      \msg_term:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2247        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2248    }
2249 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
```

```
2250     {
2251       \msg_warning:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2252         { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2253     }
```

(*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:`.

```
2254  \cs_new_protected:Nn \__enumext_check_ans_log:
2255    {
2256      \int_case:nn { \g__enumext_item_answer_diff_int }
2257        {
2258          { -1 }{ \__enumext_check_ans_log_msg_less:    }
2259          {  0 }{ \__enumext_check_ans_log_msg_same_ok: }
2260          {  1 }{ \__enumext_check_ans_log_msg_greater: }
2261        }
2262    }
2263  \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
2264    {
2265      \msg_log:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2266        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2267    }
2268  \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
2269    {
2270      \msg_log:nneee { enumext } { items-same-answer }  { \g__enumext_store_name_tl }
2271        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2272    }
2273  \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
2274    {
2275      \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2276        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2277    }
```

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

13.26.6 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.

```
2278  \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
2279    {
2280      \int_compare:nNnT
2281        { \g__enumext_check_starred_cmd_int } = { 0 }
2282        {
2283          \msg_warning:nnnV
2284            { enumext } { missing-starred }{ #1 } \l__enumext_check_start_line_env_tl
2285        }
2286      \int_compare:nNnT
2287        { \g__enumext_check_starred_cmd_int } > { 1 }
2288        {
2289          \msg_warning:nnnV
2290            { enumext } { many-starred }{ #1 } \l__enumext_check_start_line_env_tl
2291        }
2292      \int_gzero:N \g__enumext_check_starred_cmd_int
2293      \tl_clear:N \l__enumext_check_start_line_env_tl
2294    }
```

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

13.27 Keys and functions associated with storage

13.27.1 Keys for marks, wrapp and show

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

<div style="float:left">

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

</div>

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

```
2295 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2296   {
2297     \keys_define:nn { enumext / #1 }
2298       {
2299         mark-ans*  .tl_set:c  = { l__enumext_mark_answer_sym_#2_tl },
2300         mark-ans*  .initial:n = \textasteriskcentered,
2301         mark-ans*  .value_required:n = true,
2302         mark-pos*  .choice:,
2303         mark-pos* / left    .code:n = \str_set:cn { l__enumext_mark_position_#2_str } { l },
2304         mark-pos* / right   .code:n = \str_set:cn { l__enumext_mark_position_#2_str } { r },
2305         mark-pos* / center  .code:n = \str_set:cn { l__enumext_mark_position_#2_str } { c },
2306         mark-pos* / unknown .code:n =
2307                         \msg_error:nneee { enumext } { unknown-choice }
2308                           { mark-pos } { left, ~ right, ~ center } { \exp_not:n {##1} },
2309         mark-pos*  .initial:n = right,
2310         mark-pos*  .value_required:n = true,
2311         mark-sep*  .dim_set:c = { l__enumext_mark_sym_sep_#2_dim },
2312         mark-sep*  .value_required:n = true,
2313         wrap-ans*  .cs_set_protected:cp = { __enumext_keyans_wrapper_item_#2:n } ##1,
2314         wrap-ans*  .value_required:n = true,
2315         wrap-opt   .cs_set_protected:cp = { __enumext_keyans_wrapper_opt_#2:n } ##1,
2316         wrap-opt   .initial:n = [{##1}],
2317         wrap-opt   .value_required:n = true,
2318         save-sep   .tl_set:c = { l__enumext_store_keyans_item_opt_sep_#2_tl },
2319         save-sep   .initial:n = {, ~ },
2320         save-sep   .value_required:n = true,
2321         show-ans   .bool_set:N = \l__enumext_show_answer_bool,
2322         show-ans   .initial:n = false,
2323         show-ans   .value_required:n = true,
2324         show-pos   .bool_set:N = \l__enumext_show_position_bool,
2325         show-pos   .initial:n = false,
2326         show-pos   .value_required:n = true,
2327       }
2328   }
2329 \clist_map_inline:nn { {keyans}{v}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }
```

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

<div style="float:left">

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

</div>

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

```
2330 \cs_set_protected:Npn \__enumext_tmp:n #1
2331   {
2332     \keys_define:nn { enumext / #1 }
2333       {
2334         mark-ref  .tl_set:N   = \l__enumext_mark_ref_sym_tl,
2335         mark-ref  .initial:n  = \textreferencemark,
2336         mark-ref  .value_required:n = true,
2337         save-ref  .bool_set:N = \l__enumext_store_ref_key_bool,
2338         save-ref  .initial:n  = false,
2339         save-ref  .value_required:n = true,
2340         show-ans  .bool_set:N = \l__enumext_show_answer_bool,
2341         show-ans  .initial:n  = false,
2342         show-ans  .value_required:n = true,
2343         show-pos  .bool_set:N = \l__enumext_show_position_bool,
2344         show-pos  .initial:n  = false,
2345         show-pos  .value_required:n = true,
2346         mark-ans  .tl_set:N   = \l__enumext_mark_answer_sym_tl,
2347         mark-ans  .initial:n = \textasteriskcentered,
2348         mark-ans  .value_required:n = true,
2349         mark-sep  .dim_set:N  = \l__enumext_mark_sym_sep_dim,
2350         mark-sep  .value_required:n = true,
2351         mark-pos  .choice:,
2352         mark-pos / left    .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2353         mark-pos / right   .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2354         mark-pos / center  .code:n = \str_set:Nn \l__enumext_mark_position_str { c },
2355         mark-pos / unknown .code:n =
2356                         \msg_error:nneee { enumext } { unknown-choice }
```

```
2357                             { mark-pos } { left, ~ right, ~ center } { \exp_not:n {##1} },
2358        mark-pos  .initial:n = right,
2359        mark-pos  .value_required:n = true,
2360
2361        wrap-ans  .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
2362        wrap-ans  .initial:n =
2363              {
2364                  \fbox{\parbox[t]{\dimeval{\itemwidth -2\fboxsep -2\fboxrule}}{##1}}
2365              },
2366        wrap-ans  .value_required:n = true,
2367        mark-ans* .code:n = {
2368                         \keys_set:nn { enumext / keyans  } { mark-ans* = {##1} }
2369                         \keys_set:nn { enumext / keyans* } { mark-ans* = {##1} }
2370                     },
2371        mark-ans* .value_required:n = true,
2372        mark-pos* .code:n = {
2373                         \keys_set:nn { enumext / keyans  } { mark-pos* = {##1} }
2374                         \keys_set:nn { enumext / keyans* } { mark-pos* = {##1} }
2375                     },
2376        mark-pos* .value_required:n = true,
2377        mark-sep* .code:n = {
2378                         \keys_set:nn { enumext / keyans  } { mark-sep* = {##1} }
2379                         \keys_set:nn { enumext / keyans* } { mark-sep* = {##1} }
2380                     },
2381        mark-sep* .value_required:n = true,
2382        wrap-ans* .code:n = {
2383                         \keys_set:nn { enumext / keyans  } { wrap-ans* = {##1} }
2384                         \keys_set:nn { enumext / keyans* } { wrap-ans* = {##1} }
2385                     },
2386        wrap-ans* .value_required:n = true,
2387        wrap-opt  .code:n = {
2388                         \keys_set:nn { enumext / keyans  } { wrap-opt = {##1} }
2389                         \keys_set:nn { enumext / keyans* } { wrap-opt = {##1} }
2390                     },
2391        wrap-opt  .value_required:n = true,
2392        save-sep  .code:n = {
2393                         \keys_set:nn { enumext / keyans  } { save-sep = {##1} }
2394                         \keys_set:nn { enumext / keyans* } { save-sep = {##1} }
2395                     },
2396        save-sep  .value_required:n = true,
2397      }
2398    }
2399 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

13.27.2 Storing structure of the environments

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

__enumext_store_active_keys:n
__enumext_store_active_keys_vii:n

The functions __enumext_store_active_keys:n and __enumext_store_active_keys_vii:n will be responsible for the *"storing keys"* filtered from the *optional argument* of the environment in which the key save-ans is executed and the levels within this for the enumext and enumext* environments. We will execute this function only if the variable \l__enumext_store_save_key_X_bool is false, that is, the key store-key is not active, establishing the variable \l__enumext_store_save_key_X_tl with the filtered ⟨*keys*⟩.

```
2400 \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2401    {
2402      \bool_if:cF { l__enumext_store_save_key_ \__enumext_level: _bool }
2403        {
2404          \tl_clear:c { l__enumext_store_save_key_ \__enumext_level: _tl }
2405          \tl_set:ce
2406            { l__enumext_store_save_key_ \__enumext_level: _tl }
2407            { \__enumext_filter_save_key:n {#1} }
2408        }
2409    }
2410 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2411    {
2412      \bool_if:NF \l__enumext_store_save_key_vii_bool
2413        {
```

```
2414        \tl_clear:N \l__enumext_store_save_key_vii_tl
2415        \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2416      }
2417    }
```

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

13.27.3 Setting save-key key

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

save-key   The values set by this key passed in the *optional argument* of the enumext and enumext* environments will override the values of the \l__enumext_store_save_key_X_tl variable set by the functions __enumext_store_active_keys:n and __enumext_store_active_keys_vii:n. Now define the key save-key for all levels of enumext and enumext* environments.

```
2418  \cs_set_protected:Npn \__enumext_tmp:n #1
2419    {
2420      \keys_define:nn { enumext / enumext* }
2421        {
2422          save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2423          save-key .value_required:n = true,
2424        }
2425      \keys_define:nn { enumext / #1 }
2426        {
2427          save-key .code:n = \__enumext_parse_save_key:n {##1},
2428          save-key .value_required:n = true,
2429        }
2430    }
2431  \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   The functions __enumext_parse_save_key:n and __enumext_parse_save_key_vii:n will be respon-
__enumext_parse_save_key_vii:n   sible for *"storing keys"* in the variable \l__enumext_store_save_key_X_tl for enumext and enumext*.

```
2432  \cs_new_protected:Npn \__enumext_parse_save_key:n #1
2433    {
2434      \bool_set_true:c { l__enumext_store_save_key_ \__enumext_level: _bool }
2435      \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2436      \tl_set:ce
2437        { l__enumext_store_save_key_ \__enumext_level: _tl }
2438        { \__enumext_filter_save_key:n {#1} }
2439    }
2440  \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2441    {
2442      \bool_set_true:N \l__enumext_store_save_key_vii_bool
2443      \tl_clear:N \l__enumext_store_save_key_vii_tl
2444      \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2445    }
```

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

13.27.4 Internal functions to store optional arguments

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

```
2446  \cs_new:Npn \__enumext_filter_save_key:n #1
2447    {
2448      \use:e
2449        {
2450          \keyval_parse:NNn
2451            \__enumext_filter_save_key_key:n
2452            \__enumext_filter_save_key_pair:nn {#1}
2453        }
2454    }
```

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

```
2455  \cs_new:Npn \__enumext_filter_save_key_key:n #1
2456    {
2457      \str_case:nnF {#1}
2458        {
```

```
2459            { resume } {} { resume* } {} { no-store } {} { base-fix } {}
2460        }
2461        { , { \exp_not:n {#1} } }
2462    }
```

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

```
2463  \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2464    {
2465      \str_case:nnF {#1}
2466        {
2467          { series   } {} { resume    } {} { save-ans   } {} { save-ref  } {}
2468          { save-key } {} { check-ans } {} { show-ans   } {} { show-pos  } {}
2469          { mark-ans } {} { mark-pos  } {} { mark-sep   } {} { wrap-ans  } {}
2470          { mark-ans* } {} { mark-pos* } {} { mark-sep* } {} { wrap-ans* } {}
2471          { wrap-opt } {} { save-sep  } {} { mark-ref   } {} { mini-env  } {}
2472          { mini-sep } {} { mini-right } {} { mini-right* } {}
2473        }
2474        { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2475    }
```

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

13.27.5 Function for storing content in prop list

`\__enumext_store_addto_prop:n`
`\__enumext_store_addto_prop:V`

The function `\__enumext_store_addto_prop:n` stores the {⟨*content*⟩} in *prop list* defined by `save-ans` key. The *"stored content"* is retrieved by means of the `\getkeyans` command.

The form in which the {⟨*content*⟩} is *"stored"* in the *prop list* is {⟨*position*⟩}{⟨*content*⟩}. This function is used by `\anskey` in `enumext` and `enumext*` environments, `\item*` in `keyans` and `keyans*` environments and `\anspic*` in `keyanspic` environment.

```
2476  \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2477    {
2478      \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2479        {
2480          \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2481        }
2482        { #1 }
2483    }
2484  \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V }
```

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

13.27.6 Function for storing content in sequence

`\__enumext_store_addto_seq:n`
`\__enumext_store_addto_seq:v`
`\__enumext_store_addto_seq:V`

The function `\__enumext_store_addto_seq:n` stores the {⟨*content*⟩} in *sequence* defined by `save-ans` key. This function is used by `\anskey` in `enumext`, `\item*` in `keyans` and `\anspic` in `keyanspic`.

The form in which the {⟨*content*⟩} is stored in *sequence* is in a internal `enumext` or `enumext*` environments with the *"same structure"* in which the command was executed.
The *"stored content"* is retrieved by means of the `\printkeyans` command.

```
2485  \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2486    {
2487      \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
2488    }
2489  \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V }
```

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

13.27.7 Functions for storing structure in the sequence

`\__enumext_store_level_open:`
`\__enumext_store_level_close:`

The *"storing structure"* is handled by the functions `\__enumext_store_level_open:` and `\__enumext_-store_level_close:` which are executed per level within the `enumext` environment.

```
2490  \cs_new_protected:Nn \__enumext_store_level_open:
2491    {
2492      \bool_if:NT \l__enumext_check_answers_bool
2493        {
2494          \tl_if_empty:cTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2495            {
2496              \__enumext_store_addto_seq:n
2497                {
2498                  \item \begin{enumext}
```

```
2499                    }
2500                }
2501                {
2502                  \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2503                    {
2504                      \item \begin{enumext} [
2505                    }
2506                  \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2507                    {
2508                      ]
2509                    }
2510                  \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2511                }
2512            }
2513        }
2514  \cs_new_protected:Nn \__enumext_store_level_close:
2515    {
2516      \bool_if:NT \l__enumext_check_answers_bool
2517        {
2518          \__enumext_store_addto_seq:n { \end{enumext} }
2519        }
2520    }
```

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

__enumext_store_level_open_vii:
__enumext_store_level_close_vii:

The *"storing structure"* is handled by the functions __enumext_store_level_open_vii: and __enumext_-
store_level_close_vii: which are executed in the enumext* environment.

```
2521  \cs_new_protected:Nn \__enumext_store_level_open_vii:
2522    {
2523      \bool_if:NT \l__enumext_check_answers_bool
2524        {
2525          \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2526            {
2527              \__enumext_store_addto_seq:n
2528                {
2529                  \item \begin{enumext*}
2530                }
2531            }
2532            {
2533              \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2534                {
2535                  \item \begin{enumext*}[
2536                }
2537              \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2538                {
2539                  ]
2540                }
2541              \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2542            }
2543        }
2544    }
2545  \cs_new_protected:Nn \__enumext_store_level_close_vii:
2546    {
2547      \bool_if:NT \l__enumext_check_answers_bool
2548        {
2549          \__enumext_store_addto_seq:n { \end{enumext*} }
2550        }
2551    }
```

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

13.27.8 Function for show marks and position

__enumext_print_keyans_box:NN
__enumext_print_keyans_box:cc

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

#1: \l__enumext_labelwidth_X_dim
#2: \l__enumext_labelsep_X_dim

```
2552  \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2553    {
2554      \mode_leave_vertical:
2555      \skip_horizontal:n { -\dim_use:N #2 }
```

```
2556        \hbox_overlap_left:n
2557          {
2558            \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2559              {
2560                \tl_use:N \l__enumext_mark_answer_sym_tl
2561              }
2562          }
2563        \skip_horizontal:n { \dim_use:N #2 }
2564      }
2565   \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }
```

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

13.28   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.

```
2566   \cs_new_protected:Nn \__enumext_store_internal_ref:
2567     {
2568       \cs_set_protected:Npn \__enumext_tmp:n ##1
2569         {
2570           \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2571           \tl_reverse:c { l__enumext_label_copy_##1_tl }
2572           \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2573           \tl_reverse:c { l__enumext_label_copy_##1_tl }
2574         }
2575       \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2576       \cs_set:Npn \__enumext_tmp:n ##1
2577         { . \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.

```
2578         \bool_lazy_all:nT
2579           {
2580             { \bool_if_p:N \g__enumext_starred_bool }
2581             { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2582           }
2583           {
2584             \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2585               { \tl_use:N \l__enumext_label_copy_vii_tl }
2586           }
2587         \bool_lazy_all:nT
2588           {
2589             { \bool_not_p:n { \g__enumext_standar_bool } }
2590             { \bool_if_p:N \l__enumext_standar_bool }
2591             { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2592           }
2593           {
2594             \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2595               {
2596                 \tl_use:N \l__enumext_label_copy_vii_tl
2597                 \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2598               }
2599           }
```

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.

```
2600         \bool_lazy_all:nT
2601           {
2602             { \bool_if_p:N \g__enumext_standar_bool }
2603             { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2604             { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2605           }
2606           {
2607             \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2608               {
2609                 \tl_use:N \l__enumext_label_copy_i_tl
2610                 \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
```

```
2611                }
2612            }
2613        \cs_set:Npn \__enumext_tmp:n ##1
2614            { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } . }
2615        \bool_lazy_all:nT
2616            {
2617                { \bool_if_p:N \g__enumext_standar_bool }
2618                { \bool_if_p:N \l__enumext_starred_bool }
2619                { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2620            }
2621            {
2622                \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2623                    {
2624                        \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2625                        \tl_use:N \l__enumext_label_copy_vii_tl
2626                    }
2627            }
```

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

```
2628        \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2629            {
2630                \l__enumext_store_name_tl \c_colon_str
2631                \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2632            }
```

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.

```
2633        \tl_put_right:Ne \l__enumext_write_aux_file_tl
2634            {
2635                \__enumext_newlabel:nn
2636                    { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2637                    { \l__enumext_newlabel_arg_two_tl }
2638            }
2639        \l__enumext_write_aux_file_tl
2640    }
```

(*End of definition for* __enumext_store_internal_ref:.)

13.29   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*⟩}.

```
2641    \cs_new_protected:Npn \__enumext_store_anskey_code:n #1
2642        {
2643            \int_gincr:N \g__enumext_item_anskey_int
2644            \__enumext_store_addto_prop:n {#1}
2645            \bool_if:NT \l__enumext_store_ref_key_bool
2646                {
2647                    \__enumext_store_internal_ref:
2648                }
2649            \__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.

```
2650        \tl_clear:N \l__enumext_store_anskey_arg_tl
2651        \bool_lazy_and:nnT
2652            { \bool_if_p:N \l__enumext_store_columns_break_bool }
2653            { \bool_not_p:n { \l__enumext_starred_bool } }
2654            {
2655                \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2656            }
2657        \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.

```
2658        \bool_lazy_and:nnT
2659            { \bool_not_p:n { \l__enumext_starred_bool } }
2660            { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2661            {
```

```
2662        \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2663          {
2664            ( \exp_not:V \l__enumext_store_item_join_int )
2665          }
2666        }
```

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*.

```
2667        \bool_if:NTF \l__enumext_store_item_star_bool
2668          {
2669            \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2670            \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2671              {
2672                \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2673                  {
2674                    [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2675                  }
2676              }
2677            \dim_compare:nT
2678              {
2679                \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2680              }
2681              {
2682                \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2683                  {
2684                    [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2685                  }
2686              }
2687            \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2688          }
2689          {
2690            \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2691          }
```

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*.

```
2692        \bool_lazy_and:nnT
2693          { \bool_if_p:N \l__enumext_store_ref_key_bool }
2694          { \bool_if_p:N \l__enumext_hyperref_bool }
2695          {
2696            \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2697              {
2698                \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2699                  { \exp_not:V \l__enumext_mark_ref_sym_tl }
2700              }
2701          }
2702        \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2703      }
```

(*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 and wrap-sep keys.

```
2704  \cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
2705    {
2706      \par
2707      \bool_if:NTF \l__enumext_starred_bool
2708        {
2709          \dim_compare:nNnT { \l__enumext_mark_sym_sep_dim } = { \c_zero_dim }
2710            {
2711              \dim_set:Nn \l__enumext_mark_sym_sep_dim { \l__enumext_labelsep_vii_dim }
2712            }
2713          \__enumext_print_keyans_box:NN
2714            \l__enumext_labelwidth_vii_dim \l__enumext_mark_sym_sep_dim
2715        }
2716        {
2717          \dim_compare:nNnT { \l__enumext_mark_sym_sep_dim } = { \c_zero_dim }
2718            {
2719              \dim_set:Nn \l__enumext_mark_sym_sep_dim
2720                {
2721                  \dim_use:c {l__enumext_labelsep_ \__enumext_level: _dim }
2722                }
```

```
2723            }
2724          \__enumext_print_keyans_box:cc
2725            { l__enumext_labelwidth_ \__enumext_level: _dim } { l__enumext_mark_sym_sep_dim }
2726        }
2727      \__enumext_anskey_wrapper:n { #1 }
2728    }
```

(*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.

```
2729  \cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
2730    {
2731      \bool_if:NT \l__enumext_show_answer_bool
2732        {
2733          \__enumext_anskey_show_wrap_arg:n { #1 }
2734        }
2735      \bool_if:NT \l__enumext_show_position_bool
2736        {
2737          \tl_set:Ne \l__enumext_mark_answer_sym_tl
2738            {
2739              \group_begin:
2740              \exp_not:N \normalfont
2741              \exp_not:N \footnotesize [ \int_eval:n
2742                {
2743                  \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2744                }
2745                ]
2746              \group_end:
2747            }
2748          \__enumext_anskey_show_wrap_arg:n { #1 }
2749        }
2750    }
```

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

13.30   The command \anskey

Since we will be *"storing content"* in a list environment within *sequences* and can (more or less) manage the options passed to each level, it is necessary that we have a little more control over \item when storing.

The \anskey command will cover this point and give it similar behaviour to that of \item in the enumext and enumext* environments executed as follows \anskey[⟨*key = val*⟩]{⟨*content*⟩}.

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

```
2751  \keys_define:nn { enumext / anskey }
2752    {
2753      break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2754      break-col .default:n  = true,
2755      break-col .value_forbidden:n = true,
2756      item-join .int_set:N  = \l__enumext_store_item_join_int,
2757      item-join .value_required:n = true,
2758      item-star .bool_set:N = \l__enumext_store_item_star_bool,
2759      item-star .default:n  = true,
2760      item-star .value_forbidden:n = true,
2761      item-sym* .tl_set:N   = \l__enumext_store_item_symbol_tl,
2762      item-sym* .value_required:n = true,
2763      item-pos* .dim_set:N  = \l__enumext_store_item_symbol_sep_dim,
2764      item-pos* .value_required:n = true,
2765      unknown   .code:n     = { \__enumext_anskey_unknown:n {#1} },
2766    }
```

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.

```
2767  \cs_new_protected:Npn \__enumext_anskey_unknown:n #1
2768    {
2769      \exp_args:NV \__enumext_anskey_unknown:nn \l_keys_key_str {#1}
2770    }
2771  \cs_new_protected:Npn \__enumext_anskey_unknown:nn #1 #2
2772    {
2773      \tl_if_blank:nTF {#2}
```

```
2774            {
2775               \msg_error:nnn { enumext } { anskey-cmd-key-unknown } {#1}
2776            }
2777            {
2778               \msg_error:nnnn { enumext } { anskey-cmd-key-value-unknown } {#1} {#2}
2779            }
2780         }
```

(*End of definition for* break-col *and others.*)

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

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

```
2781  \NewDocumentCommand \anskey { o +m }
2782    {
2783      \__enumext_anskey_safe_outer:
2784      \group_begin:
2785        \bool_if:NT \l__enumext_check_answers_bool
2786          {
2787            \tl_if_novalue:nF {#1}
2788              {
2789                \keys_set:nn { enumext / anskey } {#1}
2790              }
2791            \tl_if_blank:nTF {#2}
2792              {
2793                \msg_error:nn { enumext } { anskey-empty-arg }
2794              }
2795              {
2796                \__enumext_anskey_safe_inner:
2797                \__enumext_store_anskey_code:n {#2}
2798              }
2799          }
2800      \group_end:
2801    }
```

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

13.30.1 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.

```
2802  \cs_new_protected:Nn \__enumext_anskey_safe_outer:
2803    {
2804      \bool_if:NF \l__enumext_store_active_bool
2805        {
2806          \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
2807        }
2808      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2809        {
2810          \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
2811        }
2812      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2813        {
2814          \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans* }
2815        }
2816      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2817        {
2818          \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
2819        }
2820    }
```

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.

```
2821  \cs_new_protected:Nn \__enumext_anskey_safe_inner:
2822    {
2823      \int_incr:N \l__enumext_anskey_level_int
2824      \int_compare:nNnT { \l__enumext_anskey_level_int } > { 1 }
```

```
2825      {
2826          \msg_error:nn { enumext } { anskey-nested }
2827      }
2828      \bool_if:NF \l__enumext_item_number_bool
2829      {
2830          \msg_error:nn { enumext } { anskey-unnumber-item }
2831      }
2832      \mode_if_math:T
2833      {
2834          \msg_error:nne { enumext } { anskey-math-mode } { \c_backslash_str anskey }
2835      }
2836  }
```

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

13.31 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: (§13.32) which is executed after the environment in which the key save-ans is active.

```
2837  \cs_new_protected:Nn \__enumext_undefine_anskey_env:
2838  {
2839      \cs_undefine:c { anskey* }
2840      \cs_undefine:c { endanskey* }
2841      \cs_undefine:c { __scontents_anskey*_env_begin: }
2842      \cs_undefine:c { __scontents_anskey*_env_end: }
2843  }
```

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

```
2844  \__enumext_before_env:nn { enumext }
2845  {
2846      \bool_lazy_and:nnT
2847          { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2848          { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2849          {
2850              \cs_if_free:cF { __scontents_anskey*_env_begin: }
2851              {
2852                  \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2853              }
2854          }
2855  }
2856  \__enumext_before_env:nn { enumext* }
2857  {
2858      \bool_lazy_and:nnT
2859          { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2860          { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2861          {
2862              \cs_if_free:cF { __scontents_anskey*_env_begin: }
2863              {
2864                  \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2865              }
2866          }
2867  }
```

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.

```
2868  \__enumext_before_env:nn { anskey* }
```

```
2869        {
2870          \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2871            {
2872              \msg_error:nnn { enumext } { anskey-env-wrong }{ keyans }
2873            }
2874          \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2875            {
2876              \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
2877            }
2878          \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2879            {
2880              \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
2881            }
2882          \bool_if:NF \l__enumext_item_number_bool
2883            {
2884              \msg_error:nn { enumext } { anskey-unnumber-item }
2885            }
2886          \mode_if_math:T
2887            {
2888              \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
2889            }
2890        }
```

(*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_anskey_env_unknown:n
__enumext_anskey_env_unknown:nn
__enumext_anskey_env_reset_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: (§13.26.1) 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.

```
2891  \cs_new_protected:Npn \__enumext_anskey_env_make:n #1
2892    {
2893      \bool_if:NT \l__enumext_anskey_env_bool
2894        {
2895          \newenvsc{anskey*}[store-env=#1,print-env=false]
2896          \__enumext_anskey_env_exec:
2897        }
2898    }
2899  \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.

```
2900  \cs_new_protected:Nn \__enumext_anskey_env_define_keys:
2901    {
2902      \keys_define:nn { scontents / scontents }
2903        {
2904          break-col .bool_gset:N = \g__enumext_store_columns_break_bool,
2905          break-col .default:n   = true,
2906          break-col .value_forbidden:n = true,
2907          item-join .int_gset:N  = \g__enumext_store_item_join_int,
2908          item-join .value_required:n = true,
2909          item-star .bool_gset:N = \g__enumext_store_item_star_bool,
2910          item-star .default:n   = true,
2911          item-star .value_forbidden:n = true,
2912          item-sym* .tl_gset:N   = \g__enumext_store_item_symbol_tl,
2913          item-sym* .value_required:n = true,
2914          item-pos* .dim_gset:N  = \g__enumext_store_item_symbol_sep_dim,
2915          item-pos* .value_required:n = true,
2916          print-env .undefine:,
2917          store-env .undefine:,
2918          write-out .undefine:,
2919          unknown   .code:n      = { \__enumext_anskey_env_unknown:n {##1} },
2920        }
2921    }
```

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.

```
2922  \cs_new_protected:Npn \__enumext_anskey_env_unknown:n #1
```

```
2923    {
2924      \exp_args:NV \__enumext_anskey_env_unknown:nn \l_keys_key_str {#1}
2925    }
2926 \cs_new_protected:Npn \__enumext_anskey_env_unknown:nn #1#2
2927    {
2928      \tl_if_blank:nTF {#2}
2929        {
2930          \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
2931        }
2932        {
2933          \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
2934        }
2935    }
```

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`.

```
2936 \cs_new_protected:Nn \__enumext_anskey_env_reset_keys:
2937    {
2938      \keys_define:nn { scontents / scontents }
2939        {
2940          break-col .undefine:,
2941          item-join .undefine:,
2942          item-star .undefine:,
2943          item-sym* .undefine:,
2944          item-pos* .undefine:,
2945          write-out .code:n     = {
2946                                  \bool_set_false:N \l__scontents_storing_bool
2947                                  \bool_set_true:N  \l__scontents_writing_bool
2948                                  \tl_set:Nn \l__scontents_fname_out_tl {##1}
2949                                 },
2950          write-out .value_required:n = true,
2951          print-env .meta:nn   = { scontents } { print-env = ##1 },
2952          print-env .default:n = true,
2953          store-env .meta:nn   = { scontents } { store-env = ##1 },
2954          unknown   .code:n    = { \__scontents_parse_environment_keys:n {##1} },
2955        }
2956    }
```

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*.

```
2957 \cs_new_protected:Npn \__enumext_rescan_anskey_env:n #1
2958    {
2959      \group_begin:
2960        \int_set:Nn \tex_newlinechar:D { `\^^J }
2961        \__scontents_rescan_tokens:x
2962          {
2963            \endgroup % This assumes \catcode`\\=0... Things might go off otherwise.
2964            #1
2965          }
2966    }
```

(*End of definition for* anskey* *and others. This function is documented on page* 14.)

`\__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*⟩.

```
2967 \cs_new_protected:Nn \__enumext_anskey_env_exec:
2968    {
2969      \__enumext_before_env:nn { anskey* }
2970        {
2971          \__enumext_anskey_env_define_keys:
2972        }
```

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.

```
2973      \hook_if_empty:nF {env/anskey*/after}
2974        {
2975          \hook_gremove_code:nn {env/anskey*/after} { * }
2976        }
2977      \__enumext_after_env:nn { anskey* }
```

```
2978            {
2979                \__enumext_anskey_env_save_keys:
2980                \tl_clear:N \l__enumext_store_anskey_env_tl
2981                \tl_clear:N \l__enumext_store_anskey_opt_tl
2982                \bool_if:NT \l__enumext_check_answers_bool
2983                  {
2984                    \tl_set:Ne \l__enumext_store_anskey_env_tl
2985                      {
2986                        \seq_item:ce { g__scontents_name_ \l__enumext_store_name_tl _seq } { -1 }
2987                      }
2988                    \regex_match:nVTF
2989                      { ^\s* \z | ^\s* \u{c__scontents_hidden_space_str} \z }
2990                      \l__enumext_store_anskey_env_tl
2991                      {
2992                        \msg_error:nn { enumext } { anskey-empty-arg }
2993                      }
2994                      {
2995                        \__enumext_anskey_env_store:
2996                      }
2997                  }
2998                \__enumext_anskey_env_clean_vars:
2999                \__enumext_anskey_env_reset_keys:
3000            }
3001        }
```

📍 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.

```
3002    \cs_new_protected:Nn \__enumext_anskey_env_save_keys:
3003      {
3004        \bool_lazy_and:nnT
3005          { \bool_if_p:N \g__enumext_store_columns_break_bool }
3006          { \bool_not_p:n { \l__enumext_starred_bool } }
3007          {
3008            \tl_put_left:Ne \l__enumext_store_anskey_opt_tl { ,break-col, }
3009          }
```

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

```
3010        \bool_lazy_and:nnT
3011          { \bool_not_p:n { \l__enumext_starred_bool } }
3012          { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }
3013          {
3014            \tl_put_left::Ne \l__enumext_store_anskey_opt_tl
3015              {
3016                ,item-join = \exp_not:V \g__enumext_store_item_join_int,
3017              }
3018          }
```

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

```
3019        \bool_if:NT \g__enumext_store_item_star_bool
3020          {
3021            \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
3022              {
3023                ,item-star,
3024              }
3025            \tl_if_empty:NF \g__enumext_store_item_symbol_tl
3026              {
3027                \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
3028                  {
3029                    ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,
3030                  }
3031              }
3032        \dim_compare:nT
3033          {
```

```
3034                  \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
3035                }
3036              {
3037                \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
3038                  {
3039                    ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
3040                  }
3041              }
3042          }
3043      }
```

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`.

```
3044  \cs_new_protected:Nn \__enumext_anskey_env_store:
3045    {
3046      \group_begin:
3047        \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
3048          {
3049            \exp_args:Ne
3050              \__enumext_store_anskey_code:n
3051                {
3052                  \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
3053                }
3054          }
3055          {
3056            \keys_set_known:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
3057            \exp_args:Ne
3058              \__enumext_store_anskey_code:n
3059                {
3060                  \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
3061                }
3062          }
3063      \group_end:
3064    }
```

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

```
3065  \cs_new_protected:Nn \__enumext_anskey_env_clean_vars:
3066    {
3067      \bool_gset_false:N \g__enumext_store_columns_break_bool
3068      \int_gzero:N        \g__enumext_store_item_join_int
3069      \bool_gset_false:N \g__enumext_store_item_star_bool
3070      \tl_gclear:N        \g__enumext_store_item_symbol_tl
3071      \dim_gzero:N        \g__enumext_store_item_symbol_sep_dim
3072    }
```

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

13.32   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* (§13.31) 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.

```
3073  \cs_new_protected:Nn \__enumext_execute_after_env:
3074    {
3075      \int_compare:nNnT { \l__enumext_level_int } = { 0 }
3076        {
3077          \tl_if_empty:NF \g__enumext_store_name_tl
3078            {
3079              \__enumext_stop_save_ans_msg:
3080              \__enumext_item_answer_diff:
3081              \__enumext_log_global_vars:
3082              \__enumext_log_answer_vars:
3083              \bool_if:NTF \g__enumext_check_ans_key_bool
3084                {
3085                  \__enumext_check_ans_show:
```

```
3086                    }
3087                  { \__enumext_check_ans_log: }
3088                \__enumext_undefine_anskey_env:
3089              }
3090            \__enumext_reset_global_vars:
3091          }
3092      }
```

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

🍃 This function is passed to the function __enumext_after_env:nn for the environments enumext (§13.39) and enumext* (§13.44) and it is executed only when the environments are not nested or at some level of these..

13.33   Common functions for keyans, keyans* and keyanspic

13.33.1   Storing content in prop list

__enumext_keyans_addto_prop:n

The function __enumext_keyans_addto_prop:n will pass the the current ⟨*label*⟩ for \item* in keyans environment and the current ⟨*label*⟩ for \anspic* in keyanspic environment followed by the ⟨*contents*⟩ of the *optional argument* of both commands to the \l__enumext_store_current_label_tl variable, which will be stored to the *prop list* defined by the save-ans key using the function __enumext_store_addto_-prop:V.

```
3093  \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
3094    {
3095      \tl_clear:N \l__enumext_store_current_label_tl
3096      \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
3097        {
3098          \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
3099        }
3100        {
3101          \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
3102        }
```

If the *optional argument* is present and the save-sep key is not empty, we save it.

```
3103      \tl_if_novalue:nF { #1 }
3104        {
3105          \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_v_tl
3106            {
3107              \tl_put_right:Nn \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_o
3108            }
3109          \tl_put_right:Nn \l__enumext_store_current_label_tl { #1 }
3110        }
3111      \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
3112    }
```

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

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

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

__enumext_keyans_store_ref:
__enumext_keyans_store_ref_aux_i:
__enumext_keyans_store_ref_aux_ii:

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

```
3113  \cs_new_protected:Nn \__enumext_keyans_store_ref:
3114    {
3115      \bool_if:NT \l__enumext_store_ref_key_bool
3116        {
3117          \cs_set_protected:Npn \__enumext_tmp:n ##1
3118            {
3119              \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
3120              \tl_reverse:c { l__enumext_label_copy_##1_tl }
3121              \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
3122              \tl_reverse:c { l__enumext_label_copy_##1_tl }
3123            }
3124          \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
3125          \__enumext_keyans_store_ref_aux_i:
3126        }
3127      }
```

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`.

```
3128  \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
3129    {
3130      \bool_if:NT \g__enumext_starred_bool
3131        {
3132          \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
3133        }
3134      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
3135        {
3136          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
3137            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
3138        }
3139      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
3140        {
3141          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
3142            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
3143        }
3144      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
3145        {
3146          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
3147            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
3148        }
3149      \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
3150        {
3151          \l__enumext_store_name_tl \c_colon_str
3152          \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
3153        }
3154      \__enumext_keyans_store_ref_aux_ii:
3155    }
```

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.

```
3156  \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
3157    {
3158      \tl_put_right:Ne \l__enumext_write_aux_file_tl
3159        {
3160          \__enumext_newlabel:nn
3161            { \exp_not:V \l__enumext_newlabel_arg_one_tl }
3162            { \l__enumext_newlabel_arg_two_tl }
3163        }
3164      \l__enumext_write_aux_file_tl
3165    }
```

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

13.33.3   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.

```
3166  \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
3167    {
3168      \tl_clear:N \l__enumext_store_current_label_tl
3169      \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
3170        {
3171          \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
3172        }
3173        {
3174          \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
3175        }
3176      \tl_if_novalue:nF { #1 }
3177        {
3178          \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_v_tl
3179            {
3180              \tl_put_right:Nn \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_o|
3181            }
3182          \tl_put_right:Nn \l__enumext_store_current_label_tl { #1 }
```

```
3183          }
3184      \__enumext_keyans_addto_seq_link:
3185  }
```

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.

```
3186  \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
3187    {
3188      \bool_lazy_and:nnT
3189          { \bool_if_p:N \l__enumext_store_ref_key_bool }
3190          { \bool_if_p:N \l__enumext_hyperref_bool }
3191          {
3192            \tl_put_right:Ne \l__enumext_store_current_label_tl
3193              {
3194                \hfill \exp_not:N \hyperlink
3195                  {
3196                    \exp_not:V \l__enumext_newlabel_arg_one_tl
3197                  }
3198                  { \exp_not:V \l__enumext_mark_ref_sym_tl }
3199              }
3200          }
3201      \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
3202      \bool_if:NT \l__enumext_check_answers_bool
3203          {
3204            \int_gincr:N \g__enumext_item_anskey_int
3205          }
3206    }
```

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

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

__enumext_keyans_save_item_opt:n
__enumext_keyans_show_item_opt:
__enumext_keyans_show_item_opt_viii:

The function __enumext_keyans_save_item_opt:n will save the optional argument of \item* and \anspic* in the variable \l__enumext_store_current_opt_arg_tl.

```
3207  \cs_new_protected:Npn \__enumext_keyans_save_item_opt:n #1
3208    {
3209      \tl_if_novalue:nF { #1 }
3210          {
3211            \tl_set:Nn \l__enumext_store_current_opt_arg_tl { #1 }
3212          }
3213    }
```

The function __enumext_keyans_show_item_opt: will print the optional arguments of \item* and \anspic* when the show-ans or show-pos keys are set next to the key wrap-opt in keyans and keyanspic environments.

```
3214  \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
3215    {
3216      \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
3217          {
3218            \bool_lazy_or:nnT
3219              { \bool_if_p:N \l__enumext_show_answer_bool }
3220              { \bool_if_p:N \l__enumext_show_position_bool }
3221              {
3222                \__enumext_keyans_wrapper_opt_v:n
3223                  { \l__enumext_store_current_opt_arg_tl } \c_space_tl
3224              }
3225          }
3226    }
```

The function __enumext_keyans_show_item_opt_viii: will print the optional argument of \item* when the show-ans or show-pos keys are set next to the key wrap-opt in keyans* environment.

```
3227  \cs_new_protected:Nn \__enumext_keyans_show_item_opt_viii:
3228    {
3229      \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
3230          {
3231            \bool_lazy_or:nnT
3232              { \bool_if_p:N \l__enumext_show_answer_bool }
3233              { \bool_if_p:N \l__enumext_show_position_bool }
3234              {
```

```
3235                \__enumext_keyans_wrapper_opt_viii:n
3236                  { \l__enumext_store_current_opt_arg_tl } \c_space_tl
3237              }
3238            }
3239        }
```

(*End of definition for* __enumext_keyans_save_item_opt:n, __enumext_keyans_show_item_opt:, *and* __enumext_keyans_-
show_item_opt_viii:.)

__enumext_keyans_pos_mark_set: The function __enumext_keyans_pos_mark_set: adjusts the horizontal spaces for the mark-sep* key
__enumext_keyans_show_ans: taking into account the value of the align key and the width of ⟨*label*⟩.
__enumext_keyans_show_pos:

```
3240  \cs_new_protected:Nn \__enumext_keyans_pos_mark_set:
3241    {
3242      \__enumext_label_width_by_box:Nn
3243        \l__enumext_mark_sep_tmpa_dim { \l__enumext_label_v_tl }
3244      \str_case:Vn \l__enumext_align_label_pos_v_str
3245        {
3246          { l }
3247            {
3248              \dim_set:Nn \l__enumext_mark_sep_tmpb_dim { \c_zero_dim }
3249            }
3250          { r }
3251            {
3252              \dim_set:Nn \l__enumext_mark_sep_tmpb_dim
3253                { \l__enumext_labelwidth_v_dim - \l__enumext_mark_sep_tmpa_dim }
3254            }
3255          { c }
3256            {
3257              \dim_set:Nn \l__enumext_mark_sep_tmpb_dim
3258                { 0.5\l__enumext_labelwidth_v_dim - 0.5\l__enumext_mark_sep_tmpa_dim }
3259            }
3260        }
```

Here we set the default values for the key mark-ans*, mark-sep* and mark-pos*.

```
3261      \dim_compare:nNnT { \l__enumext_mark_sym_sep_v_dim } = { \c_zero_dim }
3262        {
3263          \dim_set:Nn \l__enumext_mark_sym_sep_v_dim { \l__enumext_labelsep_v_dim }
3264        }
3265      \tl_set_eq:NN \l__enumext_mark_answer_sym_tl \l__enumext_mark_answer_sym_v_tl
3266      \dim_add:Nn \l__enumext_mark_sym_sep_v_dim { \l__enumext_mark_sep_tmpb_dim }
3267      \str_set_eq:NN \l__enumext_mark_position_str \l__enumext_mark_position_v_str
3268    }
```

The function __enumext_keyans_show_ans: will print the ⟨*symbol*⟩ set by the mark-ans* key when the
show-ans key is active.

```
3269  \cs_new_protected:Nn \__enumext_keyans_show_ans:
3270    {
3271      \bool_lazy_all:nT
3272        {
3273          { \bool_if_p:N \l__enumext_show_answer_bool   }
3274          { \bool_if_p:N \l__enumext_item_wrap_key_bool }
3275        }
3276        {
3277          \__enumext_keyans_pos_mark_set:
3278          \__enumext_print_keyans_box:NN
3279            \l__enumext_labelwidth_v_dim \l__enumext_mark_sym_sep_v_dim
3280        }
3281    }
```

The function __enumext_keyans_show_pos: will print the ⟨*position*⟩ of the stored content in *prop list*.
Need add 1 to \g__enumext_⟨*store name*⟩_prop for keyans environment.

```
3282  \cs_new_protected:Nn \__enumext_keyans_show_pos:
3283    {
3284      \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
3285        {
3286          \int_incr:N \l__enumext_show_pos_tmp_int
3287        }
3288        {
3289          \int_zero:N \l__enumext_show_pos_tmp_int
3290        }
3291      \bool_lazy_all:nT
3292        {
```

```
3293        { \bool_if_p:N \l__enumext_show_position_bool }
3294        { \bool_if_p:N \l__enumext_item_wrap_key_bool }
3295      }
3296      {
3297        \tl_set:Ne \l__enumext_mark_answer_sym_v_tl
3298          {
3299            \group_begin:
3300              \exp_not:N \normalfont
3301              \exp_not:N \footnotesize [ \int_eval:n
3302                {
3303                  \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3304                  + \l__enumext_show_pos_tmp_int
3305                }
3306              ]
3307            \group_end:
3308          }
3309        \__enumext_keyans_pos_mark_set:
3310        \__enumext_print_keyans_box:NN
3311          \l__enumext_labelwidth_v_dim \l__enumext_mark_sym_sep_v_dim
3312      }
3313    }
```

(*End of definition for* __enumext_keyans_pos_mark_set:, __enumext_keyans_show_ans:, *and* __enumext_keyans_show_-
pos:.)

13.34   Redefining \item and \makelabel in enumext

Redefining the \item command is not as simple as I thought. This command works in conjunction with the \makelabel command so I have to redefine both of them, in addition to this, we will have to use a couple of *global* variables to pass the values from one command to the other.

When *labeling* PDF is active \makelabel is redefined as \hss #1 and the only way to get the align key to work correctly is to redefine \makelabel using \makebox. The best way to implement this is to use the conditional command \IfDocumentMetadataTF to force this redefinition and the dedicated mode-box key to manually activate it by the user.

The \item and \item[⟨*custom*⟩] commands work in the usual way on enumext and we will add \item*, \item*[⟨*symbol*⟩] and \item*[⟨*symbol*⟩][⟨*offset*⟩].

__enumext_default_item:n   First we will see if the *optional argument* is present, if it is NOT present we will check the state of the variable \l__enumext_check_answers_bool set by the key no-store, set the boolean variable \l__enumext_-wrap_label_X_bool to "true" for the key wrap-label and execute __enumext_item_std:w and the key itemindent, otherwise we will check the state of the boolean variable \l__enumext_wrap_label_opt_-X_bool set by the key wrap-label* and execute __enumext_item_std:w with the *optional argument* and the key itemindent.

```
3314  \cs_new_protected:Npn \__enumext_default_item:n #1
3315    {
3316      \tl_if_novalue:nTF {#1}
3317        {
3318          \bool_if:NT \l__enumext_check_answers_bool
3319            {
3320              \int_gincr:N \g__enumext_item_number_int
3321              \bool_set_true:N \l__enumext_item_number_bool
3322            }
3323          \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
3324          \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3325        }
3326        {
3327          \bool_set_eq:cc
3328            { l__enumext_wrap_label_ \__enumext_level: _bool }
3329            { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
3330          \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl
3331        }
3332    }
```

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

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

```
#1:  \l__enumext_item_symbol_X_tl
#2:  \l__enumext_item_symbol_sep_X_dim
```

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

```
3333  \cs_new_protected:Npn \__enumext_item_starred_exec:nn #1 #2
3334    {
3335      \tl_if_novalue:nTF {#1}
3336        {
3337          \tl_gset_eq:Nc
3338            \g__enumext_item_symbol_aux_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
3339        }
3340        {
3341          \tl_gset:Nn \g__enumext_item_symbol_aux_tl {#1}
3342        }
3343      \tl_if_novalue:nTF {#2}
3344        {
3345          \dim_set_eq:cc
3346            { l__enumext_item_symbol_sep_ \__enumext_level: _dim }
3347            { l__enumext_labelsep_ \__enumext_level: _dim }
3348        }
3349        {
3350          \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
3351        }
3352      \bool_if:NT \l__enumext_check_answers_bool
3353        {
3354          \int_gincr:N \g__enumext_item_number_int
3355          \bool_set_true:N \l__enumext_item_number_bool
3356        }
3357      \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
3358      \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3359    }
```

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

```
3360  \cs_new_protected:Nn \__enumext_item_starred_exec:
3361    {
3362      \tl_if_empty:cF { l__enumext_item_symbol_ \__enumext_level: _tl }
3363        {
3364          \mode_leave_vertical:
3365          \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3366          \hbox_overlap_left:n { \g__enumext_item_symbol_aux_tl }
3367          \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3368        }
3369    }
```

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

`\__enumext_redefine_item:` The function `\__enumext_redefine_item:` will redefine the `\item` command in the `enumext` environment adding `\item*`. This function are passed to `\__enumext_list_arg_two_X:` used in the definition of the `enumext` environment (§13.39).

```
3370  \cs_new_protected:Nn \__enumext_redefine_item:
3371    {
3372      \RenewDocumentCommand \item { s o o }
3373        {
3374          \bool_if:nTF {##1}
3375            {
3376              \__enumext_item_starred_exec:nn {##2} {##3}
3377            }
3378            { \__enumext_default_item:n {##2} }
3379        }
3380    }
```

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

`\__enumext_make_label:`
`\__enumext_make_label_std:`
`\__enumext_make_label_box:`

The function `\__enumext_make_label:` redefine `\makelabel` for the keys `mode-box`, `align`, `font`, `wrap-label`, `wrap-label*` and `\item*` for `enumext` environment. This function are passed to `\__enumext_list_arg_two_X:` used in the definition of the `enumext` environment (§13.39).

```
3381  \cs_new_protected:Nn \__enumext_make_label:
```

```
3382      {
3383        \IfDocumentMetadataTF
3384          {
3385            \__enumext_make_label_box:
3386          }
3387          {
3388            \bool_if:NTF \l__enumext_mode_box_bool
3389              {
3390                \__enumext_make_label_box:
3391              }
3392              {
3393                \__enumext_make_label_std:
3394              }
3395          }
3396      }
```

Standard definition when \DocumentMetadata is not active.

```
3397   \cs_new_protected:Nn \__enumext_make_label_std:
3398      {
3399        \RenewDocumentCommand \makelabel { m }
3400          {
3401            \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
3402            \__enumext_item_starred_exec:
3403            \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3404            \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3405              {
3406                \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3407              }
3408              { ##1 }
3409            \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
3410            \tl_gclear:N \g__enumext_item_symbol_aux_tl
3411          }
3412      }
```

Definition using \makebox when \DocumentMetadata is active or mode-box is active.

🔷 Here it is necessary to use \strut\smash to maintain text *alignment* in case the user wants to use \labelbx for example. In my experiments with *mimicking* the description environment it was the only way out and it seems to have no adverse effects and may serve in the future as a basis for a more generic list environment package than enumext.

```
3413   \cs_new_protected:Nn \__enumext_make_label_box:
3414      {
3415        \RenewDocumentCommand \makelabel { m }
3416          {
3417            \strut\smash
3418              {
3419                \makebox
3420                  [ \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim } ]
3421                  [ \str_use:c { l__enumext_align_label_pos_ \__enumext_level: _str } ]
3422                  {
3423                    \__enumext_item_starred_exec:
3424                    \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3425                    \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3426                      {
3427                        \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3428                      }
3429                      { ##1 }
3430                    \tl_gclear:N \g__enumext_item_symbol_aux_tl
3431                  }
3432              } % close smash
3433          }
3434      }
```

(*End of definition for* __enumext_make_label:, __enumext_make_label_std:, *and* __enumext_make_label_box:.)

13.35   Setting item-sym* and item-pos* keys

In order to have a cleaner implementation of \item* for the enumext and enumext* environments it is best to define a couple of keys that allow us to control and set by default the ⟨symbol⟩ and its ⟨offset⟩.

item-sym*   Define and set item-sym* and item-pos* keys for enumext and enumext*.
item-pos*
```
3435   \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
3436      {
3437        \keys_define:nn { enumext / #1 }
```

```
3438          {
3439              item-sym* .tl_set:c   = { l__enumext_item_symbol_#2_tl },
3440              item-sym* .value_required:n = true,
3441              item-sym* .initial:n  = {\textborn},
3442              item-pos* .dim_set:c  = { l__enumext_item_symbol_sep_#2_dim },
3443              item-pos* .value_required:n = true,
3444          }
3445      }
3446  \clist_map_inline:nn
3447      {
3448          {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
3449      }
3450      { \__enumext_tmp:nn #1 }
```

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

13.36 Handling unknown keys

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

☞* Well, the paragraph above is not so real, after all I had to add more ⟨*keys*⟩ than I had planned, not everything turns out the way one thinks in life.

13.36.1 Handling unknown keys for keyans, keyans* and keyanspic

unknown
__enumext_keyans_unknown_keys:n
__enumext_keyans_unknown_keys:nn

Define and set unknown key for keyans, keyans* and keyanspic environments. Here it is necessary to set \l__enumext_envir_name_tl in case an unknown key is passed using \setenumext.

```
3451  \cs_set_protected:Npn \__enumext_tmp:n #1
3452      {
3453          \keys_define:nn { enumext / #1 }
3454              {
3455                  unknown .code:n = {
3456                                      \tl_set:Nn \l__enumext_envir_name_tl {#1}
3457                                      \__enumext_keyans_unknown_keys:n {##1}
3458                                    },
3459              }
3460      }
3461  \clist_map_inline:nn { keyans, keyans*, keyanspic } { \__enumext_tmp:n {#1} }
```

Internal functions for handling unknown key.

```
3462  \cs_new_protected:Npn \__enumext_keyans_unknown_keys:n #1
3463      {
3464          \exp_args:NV \__enumext_keyans_unknown_keys:nn \l_keys_key_str {#1}
3465      }
3466  \cs_new_protected:Npn \__enumext_keyans_unknown_keys:nn #1#2
3467      {
3468          \tl_if_blank:nTF {#2}
3469              {
3470                  \msg_error:nne { enumext } { keyans-unknown-key } {#1}
3471              }
3472              {
3473                  \msg_error:nnee { enumext } { keyans-unknown-key-value } {#1} {#2}
3474              }
3475      }
```

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

13.36.2 Handling unknown keys for enumext*

unknown
__enumext_starred_unknown_keys:n
__enumext_starred_unknown_keys:nn

Define and set unknown key for enumext* environment.

```
3476  \keys_define:nn { enumext / enumext* }
3477      {
3478          unknown .code:n = { \__enumext_starred_unknown_keys:n {#1} },
3479      }
```

Internal functions for handling unknown key.

```
3480  \cs_new_protected:Npn \__enumext_starred_unknown_keys:n #1
3481      {
3482          \exp_args:NV \__enumext_starred_unknown_keys:nn \l_keys_key_str {#1}
3483      }
3484  \cs_new_protected:Npn \__enumext_starred_unknown_keys:nn #1#2
3485      {
3486          \tl_if_blank:nTF {#2}
```

```
3487        {
3488            \msg_error:nnn { enumext } { starred-unknown-key } {#1}
3489        }
3490        {
3491            \msg_error:nnnn { enumext } { starred-unknown-key-value } {#1} {#2}
3492        }
3493    }
```

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

13.36.3   Handling unknown keys for enumext

unknown
__enumext_standar_unknown_keys:n
__enumext_standar_unknown_keys:nn

Defines and set the key unknown for enumext environment.

```
3494 \cs_set_protected:Npn \__enumext_tmp:n #1
3495    {
3496        \keys_define:nn { enumext / #1 }
3497            {
3498                unknown .code:n = { \__enumext_standar_unknown_keys:n {##1} },
3499            }
3500    }
3501 \clist_map_inline:nn { level-1,level-2,level-3,level-4 } { \__enumext_tmp:n {#1} }
```

Internal functions for handling unknown key.

```
3502 \cs_new_protected:Npn \__enumext_standar_unknown_keys:n #1
3503    {
3504        \exp_args:NV \__enumext_standar_unknown_keys:nn \l_keys_key_str {#1}
3505    }
3506 \cs_new_protected:Npn \__enumext_standar_unknown_keys:nn #1#2
3507    {
3508        \tl_if_blank:nTF {#2}
3509            {
3510                \msg_error:nnn { enumext } { standar-unknown-key } {#1}
3511            }
3512            {
3513                \msg_error:nnnn { enumext } { standar-unknown-key-value } {#1} {#2}
3514            }
3515    }
```

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

13.37   Redefining \item and \makelabel in keyans

The \item and \item[⟨*custom*⟩] commands work in the usual way in keyans, but the \item* and \item*[⟨*content*⟩] commands *store* the current ⟨*label*⟩ next to the ⟨*content*⟩ if it is present in the *sequence* and *prop list* defined by save-ans key.

__enumext_keyans_default_item:n

The function __enumext_keyans_default_item:n executes the original behavior of the \item along with the keys wrap-label, wrap-label* and itemindent.

```
3516 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
3517    {
3518        \tl_if_novalue:nTF { #1 }
3519            {
3520                \bool_set_true:N \l__enumext_wrap_label_v_bool
3521                \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
3522            }
3523            {
3524                \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
3525                \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
3526            }
3527    }
```

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

__enumext_keyans_starred_item:n

The function __enumext_keyans_starred_item:n will take as argument #1 the *optional argument* [⟨*content*⟩] passed to \item* and save it via the __enumext_keyans_save_item_opt:n function, then activate the wrap-label key, execute \item using __enumext_item_std:w, the itemindent key and print the *optional argument* using the __enumext_keyans_show_item_opt: function handled by the wrap-opt key.

```
3528 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
3529    {
3530        \__enumext_keyans_save_item_opt:n { #1 }
3531        \bool_set_true:N \l__enumext_wrap_label_v_bool
3532        \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
3533        \__enumext_keyans_show_item_opt:
```

Now *store* the current ⟨*label*⟩ first in the *prop list* (including the *optional argument*), run the internal *"label and ref"* system if the `save-ref` key is active, then *store* in the *sequence* and finally increments `\g__enumext_-check_starred_cmd_int` for internal check system.

```
3534        \__enumext_keyans_addto_prop:n { #1 }
3535        \__enumext_keyans_store_ref:
3536        \__enumext_keyans_addto_seq:n { #1 }
3537        \int_gincr:N \g__enumext_check_starred_cmd_int
3538      }
```

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

`\item*`
`\__enumext_keyans_redefine_item:`

The function `\__enumext_keyans_redefine_item:` is responsible for adding the *starred argument* and *optional argument* by the `\__enumext_list_arg_two_v:` function in the definition of the `keyans` environment. Here we will set to true the variable `\l__enumext_item_wrap_key_bool` used by the `wrap-ans*` key only when `\item*` is executed and additionally we need to use `\peek_remove_spaces:n` to avoid an unwanted space when using `\item*` together with the `itemindent` key. This function are passed to `\__enumext_list_arg_two_v:` used in the definition of the `keyans` environment (§13.38).

```
3539  \cs_new_protected:Nn \__enumext_keyans_redefine_item:
3540    {
3541      \RenewDocumentCommand \item { s o }
3542        {
3543          \bool_if:nTF {##1}
3544            {
3545              \bool_set_true:N \l__enumext_item_wrap_key_bool % wrap-ans*
3546              \peek_remove_spaces:n
3547                {
3548                  \__enumext_keyans_starred_item:n {##2}
3549                }
3550            }
3551            {
3552              \bool_set_false:N \l__enumext_item_wrap_key_bool
3553              \__enumext_keyans_default_item:n {##2}
3554            }
3555        }
3556    }
```

(*End of definition for* `\item*` *and* `\__enumext_keyans_redefine_item:`. *This function is documented on page 16.*)

`\__enumext_keyans_make_label:`
`\__enumext_keyans_wrapper_label:n`
`\__enumext_keyans_make_label_std:`
`\__enumext_keyans_make_label_box:`

The function `\__enumext_keyans_make_label:` redefine `\makelabel` for the keys `mode-box`, `align`, `font`, `wrap-label`, `wrap-label*`, `wrap-ans*` and `\item*` for `keyans` environment. This function are passed to `\__enumext_list_arg_two_v:` used in the definition of the `keyans` environment (§13.38).

```
3557  \cs_new_protected:Nn \__enumext_keyans_make_label:
3558    {
3559      \IfDocumentMetadataTF
3560        {
3561          \__enumext_keyans_make_label_box:
3562        }
3563        {
3564          \bool_if:NTF \l__enumext_mode_box_bool
3565            {
3566              \__enumext_keyans_make_label_box:
3567            }
3568            {
3569              \__enumext_keyans_make_label_std:
3570            }
3571        }
3572    }
```

We added conditionals to the `\__enumext_keyans_wraper_label:n` function to handle the keys `wrap-ans*`, `wrap-label` and `wrap-label*`.

```
3573  \cs_new_protected:Npn \__enumext_keyans_wrapper_label:n #1
3574    {
3575      \bool_lazy_all:nT
3576        {
3577          { \bool_if_p:N \l__enumext_wrap_label_v_bool          }
3578          { \bool_if_p:N \l__enumext_show_answer_bool           }
3579          { \bool_if_p:N \l__enumext_item_wrap_key_bool         }
3580          { \cs_if_exist_p:N \__enumext_keyans_wrapper_item_v:n }
3581        }
3582        {
```

```
3583          \cs_set_eq:NN \__enumext_wrapper_label_v:n \__enumext_keyans_wrapper_item_v:n
3584        }
3585      \bool_if:NTF \l__enumext_wrap_label_v_bool
3586        {
3587          \__enumext_wrapper_label_v:n { #1 }
3588        }
3589        { #1 }
3590    }
```

Standard definition when \DocumentMetadata is not active.

```
3591  \cs_new_protected:Nn \__enumext_keyans_make_label_std:
3592    {
3593      \RenewDocumentCommand \makelabel { m }
3594        {
3595          \tl_use:N \l__enumext_label_fill_left_v_tl
3596          \__enumext_keyans_show_ans:
3597          \__enumext_keyans_show_pos:
3598          \tl_use:N \l__enumext_label_font_style_v_tl
3599          \__enumext_keyans_wrapper_label:n { ##1 }
3600          \tl_use:N \l__enumext_label_fill_right_v_tl
3601        }
3602    }
```

Definition using \makebox when \DocumentMetadata is active or mode-box is active.

```
3603  \cs_new_protected:Nn \__enumext_keyans_make_label_box:
3604    {
3605      \RenewDocumentCommand \makelabel { m }
3606        {
3607          \strut\smash
3608            {
3609              \makebox[ \l__enumext_labelwidth_v_dim ][ \l__enumext_align_label_pos_v_str ]
3610                {
3611                  \__enumext_keyans_show_ans:
3612                  \__enumext_keyans_show_pos:
3613                  \tl_use:N \l__enumext_label_font_style_v_tl
3614                  \__enumext_keyans_wrapper_label:n { ##1 }
3615                }
3616            }
3617        }
3618    }
```

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

13.38   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.

13.38.1   Calculation of \leftmargin and \itemindent

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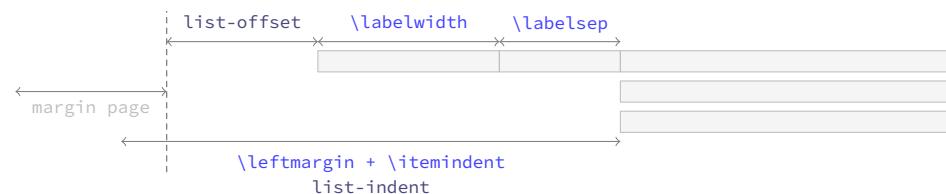
Figure 9: Representation of standard horizontal lengths in list environment.

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

__enumext_calc_hspace:NNNNNNN

__enumext_calc_hspace:ccccccc

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

```
        #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
```
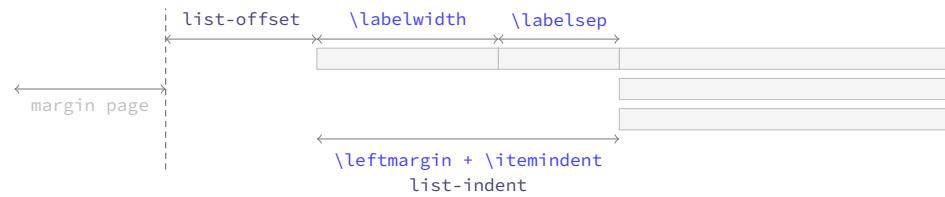
Figure 10: Representation of horizontal lengths concept in list in enumext.
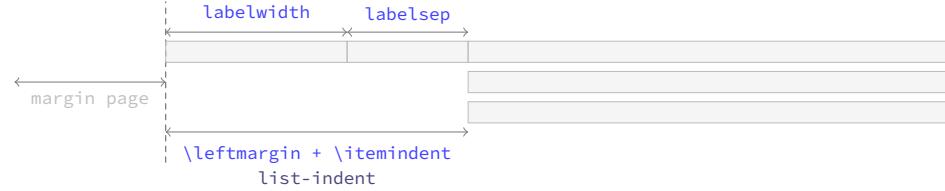
Figure 11: Default horizontal lengths in enumext.

```
#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 (§13.38).

```
3619 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
3620   {
3621     \dim_compare:nNnT { #1 } < { \c_zero_dim }
3622       {
3623         \msg_warning:nnnV { enumext } { width-non-positive }{ labelwidth }{ #1 }
3624         \dim_set:Nn #1 { \dim_abs:n { #1 } }
3625       }
3626     \dim_compare:nNnT { #2 } < { \c_zero_dim }
3627       {
3628         \msg_warning:nnnV { enumext } { width-negative }{ labelsep }{ #2 }
3629         \dim_set:Nn #2 { \dim_abs:n { #2 } }
3630       }
```

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

```
3631       \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2} }
```

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

```
3632       \dim_compare:nNnTF { #4 } < { \c_zero_dim }
3633         {
3634         \dim_set:Nn #6 { #1 + #2 - #4}
3635         \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3636         }
3637         {
3638         \dim_compare:nNnT { #4 } = { #1 + #2 }
3639           { \dim_set:Nn #6 { \c_zero_dim } }
3640         \dim_compare:nNnT { #4 } < { #1 + #2 }
3641           { \dim_set:Nn #6 { #1 + #2 - #4} }
3642         \dim_compare:nNnT { #4 } > { #1 + #2 }
3643           {
3644             \dim_set:Nn #6 { -#1 - #2 + #4}
3645             \dim_set:Nn #6 { #6*-1}
3646           }
3647         \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3648         }
3649   }
3650 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { ccccccc }
```

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

13.38.2 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.

```
3651 \cs_set_protected:Npn \__enumext_tmp:n #1
3652   {
3653     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
```

```
3654              {
3655                \__enumext_calc_hspace:ccccccc
3656                  { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3657                  { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3658                  { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3659                  { l__enumext_leftmargin_tmp_#1_bool }
3660                \clist_map_inline:nn
3661                  { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3662                  { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
3663                \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3664                  { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
3665                \usecounter { enumX#1 }
3666                \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
3667                \str_if_eq:nnTF {#1} { v }
3668                  {
3669                    \__enumext_keyans_redefine_item:
3670                    \__enumext_keyans_make_label:
3671                    \__enumext_keyans_ref:
3672                    \__enumext_keyans_fake_item_indent:
3673                    \bool_if:cT { l__enumext_show_length_#1_bool }
3674                      {
3675                        \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3676                      }
3677                  }
3678                  {
3679                    \__enumext_redefine_item:
3680                    \__enumext_make_label:
3681                    \__enumext_standar_ref:
3682                    \__enumext_fake_item_indent:
3683                    \bool_if:cT { l__enumext_show_length_#1_bool }
3684                      {
3685                        \msg_term:nnne { enumext } { list-lengths } {#1}
3686                          { \int_use:N \l__enumext_level_int }
3687                      }
3688                  }
3689              }
3690          }
3691      \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }
```

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

`\__enumext_list_arg_two_vii:`
`\__enumext_list_arg_two_viii:`

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

```
3692  \cs_set_protected:Npn \__enumext_tmp:n #1
3693    {
3694      \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3695        {
3696          \bool_set_true:c { l__enumext_leftmargin_tmp_#1_bool }
3697          \dim_zero:c { l__enumext_leftmargin_tmp_#1_dim }
3698          \__enumext_calc_hspace:ccccccc
3699            { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3700            { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3701            { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3702            { l__enumext_leftmargin_tmp_#1_bool }
3703          \clist_map_inline:nn
3704            { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3705            { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
3706          \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3707            { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
3708          \skip_set_eq:Nc \parsep  { l__enumext_itemsep_#1_skip }
3709          \skip_zero:N \partopsep
3710          \usecounter { enumX#1 }
3711          \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
3712          \__enumext_starred_ref:
3713          \str_if_eq:nnTF {#1} { vii }
3714            {
3715              \__enumext_fake_item_indent_vii:
3716              \bool_if:cT { l__enumext_show_length_vii_bool }
3717                { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
```

```
3718              }
3719              {
3720                \__enumext_fake_item_indent_viii:
3721                \bool_if:cT { l__enumext_show_length_#1_bool }
3722                  { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3723              }
3724          }
3725        }
3726      \clist_map_inline:nn { vii, viii } { \__enumext_tmp:n {#1} }
```

(*End of definition for* __enumext_list_arg_two_vii: *and* __enumext_list_arg_two_viii:.)

13.39   The environment enumext

__enumext_safe_exec: The __enumext_safe_exec: function first call the function __enumext_is_not_nested: which sets \g__enumext_standar_bool to *"true"* if we are NOT nested within enumext*, then call the function __enumext_internal_mini_page: to create the environment __enumext_mini_page, we will increment \l__enumext_level_int to restrict nesting of the environment, set \l__enumext_standar_bool to *"true"* and finally call the function __enumext_is_on_first_level: which sets \l__enumext_standar_-first_bool to *"true"* only if the environment is NOT nested and we are at the *"first level"*.

```
3727  \cs_new_protected:Nn \__enumext_safe_exec:
3728    {
3729      \__enumext_is_not_nested:
3730      \__enumext_internal_mini_page:
3731      \int_incr:N \l__enumext_level_int
3732      \int_compare:nNnT { \l__enumext_level_int } > { 4 }
3733        { \msg_fatal:nn { enumext } { list-too-deep } }
3734      \bool_set_true:N \l__enumext_standar_bool
3735      \bool_set_false:N \l__enumext_starred_bool
3736      \__enumext_is_on_first_level:
3737    }
```

(*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 *sequence* if the key save-key is not active.

```
3738  \cs_new_protected:Npn \__enumext_parse_keys:n #1
3739    {
3740      \tl_if_novalue:nF {#1}
3741        {
3742          \str_clear:N \l__enumext_series_str
3743          \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
3744            {
3745              \keys_set:nn { enumext / level-1 } {#1}
3746              \__enumext_parse_series:n {#1}
3747              \__enumext_nested_base_line_fix:
3748            }
3749            {
3750              \exp_args:Ne \keys_set:nn
3751                { enumext / level-\int_use:N \l__enumext_level_int } {#1}
3752            }
3753          \__enumext_store_active_keys:n {#1}
3754        }
3755    }
```

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

__enumext_start_store_level: The __enumext_start_store_level: function activate the *"storing structure"* mechanism in the *sequence* for the command \anskey and the environment anskey*.

```
3756  \cs_new_protected:Nn \__enumext_start_store_level:
3757    {
3758      \bool_lazy_all:nT
3759        {
3760          { \bool_if_p:N \l__enumext_store_active_bool }
3761          { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3762          { \bool_if_p:N \g__enumext_standar_bool }
3763        }
```

```
3764            {
3765                \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3766                  {
3767                    \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3768                    \__enumext_store_level_open:
3769                  }
3770              }
```

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

```
3771            \bool_lazy_all:nT
3772              {
3773                { \bool_if_p:N \l__enumext_store_active_bool }
3774                { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3775                { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
3776              }
3777              {
3778                \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3779                  {
3780                    \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3781                    \__enumext_store_level_open:
3782                  }
3783              }
3784          }
```

(*End of definition for* __enumext_start_store_level:.)

__enumext_stop_store_level: The __enumext_stop_store_level: function stop the *"storing structure"* mechanism in the *sequence* for the command \anskey and the environment anskey*.

```
3785  \cs_new_protected:Nn \__enumext_stop_store_level:
3786    {
3787      \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
3788        {
3789            \__enumext_store_level_close:
3790        }
3791    }
```

(*End of definition for* __enumext_stop_store_level:.)

__enumext_multicols_start: The function __enumext_multicols_start: will start the multicols environment according to the value passed by the columns key, then set the default value for \columnsep when columns-sep=0pt and set the value of \multicolsep equal to zero and leave \columnseprule equal to zero for inner levels.

```
3792  \cs_new_protected:Nn \__enumext_multicols_start:
3793    {
3794      \int_compare:nNnT
3795        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3796        {
3797          \dim_compare:nNnT
3798            { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3799            {
3800              \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
3801                {
3802                  ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
3803                    + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
3804                  ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
3805                  - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3806                }
3807            }
3808          \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim  }
3809          \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3810            {
3811              \dim_zero:N \columnseprule
3812            }
```

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.

```
3813            \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3814              {
3815                \skip_zero:N \multicolsep
3816                \__enumext_multi_addvspace:
3817              }
3818            \raggedcolumns
3819            \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
```

```
3820          }
3821       }
```

(*End of definition for* __enumext_multicols_start:.)

__enumext_multicols_stop: The function __enumext_multicols_stop: will stop the multicols environment and apply our *"vertical adjust"* spacing. For compatibility with *tagged* PDF, the closing of the list environment is executed here along with __enumext_stop_store_level:.

```
3822  \cs_new_protected:Nn \__enumext_multicols_stop:
3823    {
3824      \int_compare:nNnTF
3825        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3826        {
3827          \__enumext_stop_list:
3828          \__enumext_stop_store_level:
3829          \end{multicols}
3830          \__enumext_unskip_unkern:
3831          \__enumext_unskip_unkern:
3832          \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
3833        }
3834        {
3835          \__enumext_stop_list:
3836          \__enumext_stop_store_level:
3837        }
3838    }
```

(*End of definition for* __enumext_multicols_stop:.)

__enumext_before_list: The function __enumext_before_list: first calls the function __enumext_vspace_above: used by the keys above and above*, then calls the function __enumext_before_args_exec: used by the key before* and finally execute the function __enumext_check_ans_active: for the check answer mechanism.

```
3839  \cs_new_protected:Nn \__enumext_before_list:
3840    {
3841      \__enumext_vspace_above:
3842      \__enumext_before_args_exec:
3843      \__enumext_check_ans_active:
```

When the mini-env key is active it will set the value of the \l__enumext_minipage_right_X_dim to be the *width* of the __enumext_mini_page environment on the *"right side"*, using this value together with the value of the \l__enumext_minipage_hsep_X_dim set by the mini-sep key, the value of \l__enumext_-minipage_left_X_dim will be set, which will be the *width* of __enumext_mini_page environment on the *"left side"*, always having a current \linewidth as *maximum width* between them.

```
3844      \dim_compare:nNnT
3845        { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3846        {
3847          \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
3848            {
3849              \linewidth
3850              - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
3851              - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
3852            }
```

The boolean variable \l__enumext_minipage_active_X_bool will be activated and the integer variable \g__enumext_minipage_stat_int used by the \miniright command will be incremented, then the function __enumext_minipage_add_space: is called and the __enumext_mini_page environment on the *"left side"* will be initialized followed by the *"vertical spacing"* applied to preserve the *"baseline"* between the *left* and *right* side environments. After these actions, the function __enumext_multicols_start: is called to handle the multicols environment.

```
3853          \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
3854          \int_gincr:N \g__enumext_minipage_stat_int
3855          \__enumext_minipage_add_space:
3856          \noindent
3857          \__enumext_mini_page{ \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
3858        }
3859      \__enumext_multicols_start:
3860    }
```

(*End of definition for* __enumext_before_list:.)

`\__enumext_second_part:`

The function `\__enumext_second_part:` first check the state of the boolean variable `\l__enumext_-minipage_active_X_bool`, if it is "true" a small test will be executed to check if we have omitted the use of `\miniright` (the `__enumext_mini_page` environment has not been closed), then close `__enumext_mini_page` and add the *adjusted vertical space* `\l__enumext_minipage_after_skip`, otherwise we will close the `multicols` environment.

```
3861  \cs_new_protected:Nn \__enumext_second_part:
3862    {
3863      \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
3864        {
3865          \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3866            {
3867              \msg_warning:nn { enumext } { missing-miniright }
3868              \miniright
3869            }
3870          \int_gzero:N \g__enumext_minipage_stat_int
3871          \__enumext_unskip_unkern: % remove topsep + [partopsep]
3872          \end__enumext_mini_page
3873        }
3874        {
3875          \__enumext_multicols_stop:
3876        }
```

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.

```
3877      \__enumext_after_stop_list:
3878      \__enumext_check_ans_key_hook:
3879      \__enumext_vspace_below:
3880      \bool_set_false:N \l__enumext_standar_bool
3881      \__enumext_resume_save_counter:
3882    }
```

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

`\__enumext_set_item_width:`

The function `\__enumext_set_item_width:` will set the value of `\itemwidth` taking into account the value established by the `list-offset` key for each level of the environment.

```
3883  \cs_new_protected:Nn \__enumext_set_item_width:
3884    {
3885      \dim_set:Nn \itemwidth { \linewidth }
3886      \dim_compare:nT
3887        {
3888          \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim } != \c_zero_dim
3889        }
3890        {
3891          \dim_sub:Nn \itemwidth
3892            {
3893              \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3894            }
3895        }
3896    }
```

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

enumext

Now create the `enumext` environment based on `list` environment by levels.

```
3897  \NewDocumentEnvironment{enumext}{ O{} }
3898    {
3899      \__enumext_safe_exec:
3900      \__enumext_parse_keys:n {#1}
3901      \__enumext_before_list:
3902      \__enumext_start_store_level:
3903      \__enumext_start_list:nn
3904        { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
3905        {
3906          \use:c { __enumext_list_arg_two_ \__enumext_level: : }
3907          \__enumext_before_keys_exec:
3908        }
3909      \__enumext_set_item_width:
3910      \__enumext_after_args_exec:
3911    }
3912    {
```

```
3913        \__enumext_second_part:
3914      }
```

(*End of definition for* enumext. *This function is documented on page 5.*)

As we don't want our check to be executed check-ans by levels but on the complete list, we will take it out of the enumext environment using the *"hook"* function __enumext_after_env:nn.

```
3915  \__enumext_after_env:nn {enumext}
3916    {
3917      \__enumext_execute_after_env:
3918    }
```

13.40   The environment keyans

The environment keyans also based on lists. The main differences with the enumext environment are the *nesting* and the way the *answers* (choice) will be stored and checked, this environment is intended exclusively for *"multiple choice questions".*

__enumext_keyans_safe_exec:

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

```
3919  \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3920    {
3921      \bool_if:NF \l__enumext_store_active_bool
3922        {
3923          \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
3924        }
3925      \int_incr:N \l__enumext_keyans_level_int
3926      \bool_set_true:N \l__enumext_keyans_env_bool
3927      \__enumext_keyans_name_and_start:
3928      % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3929      \bool_set_false:N \l__enumext_store_active_bool
3930      \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
3931        {
3932          \msg_error:nn { enumext } { keyans-nested }
3933        }
3934      \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3935        {
3936          \msg_error:nn { enumext } { keyans-wrong-level }
3937        }
3938    }
```

(*End of definition for* __enumext_keyans_safe_exec:.)

__enumext_keyans_parse_keys:n

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

```
3939  \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3940    {
3941      \keys_set:nn { enumext / keyans } {#1}
3942    }
```

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

__enumext_before_list_v:
__enumext_keyans_multicols_start:
__enumext_keyans_multicols_stop:
__enumext_second_part_v:

Same implementation as the one used in the enumext environment.

```
3943  \cs_new_protected:Nn \__enumext_before_list_v:
3944    {
3945      \__enumext_vspace_above_v:
3946      \__enumext_before_args_exec_v:
3947      \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3948        {
3949          \dim_set:Nn \l__enumext_minipage_left_v_dim
3950            {
3951              \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3952            }
3953          \bool_set_true:N \l__enumext_minipage_active_v_bool
3954          \int_gincr:N \g__enumext_minipage_stat_int
3955          \__enumext_keyans_minipage_add_space:
3956          \__enumext_mini_page{ \l__enumext_minipage_left_v_dim }
3957        }
3958      \__enumext_keyans_multicols_start:
3959    }
3960  \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3961    {
```

```
3962        \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3963          {
3964            \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3965              {
3966                \dim_set:Nn \l__enumext_columns_sep_v_dim
3967                  {
3968                    (
3969                      \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3970                    ) / \l__enumext_columns_v_int
3971                    - \l__enumext_listoffset_v_dim
3972                  }
3973              }
3974            \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim
3975            \dim_zero:N \columnseprule % no rule here
3976            \bool_if:NF \l__enumext_minipage_active_v_bool
3977              {
3978                \skip_zero:N \multicolsep
3979                \__enumext_keyans_multi_addvspace:
3980              }
3981            \raggedcolumns
3982            \begin{multicols}{ \l__enumext_columns_v_int }
3983          }
3984      }
3985 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3986   {
3987     \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
3988       {
3989         \__enumext_stop_list:
3990         \end{multicols}
3991         \__enumext_unskip_unkern:
3992         \__enumext_unskip_unkern:
3993         \par\addvspace{ \l__enumext_multicols_below_v_skip }
3994       }
3995       {
3996         \__enumext_stop_list:
3997       }
3998   }
3999 \cs_new_protected:Nn \__enumext_second_part_v:
4000   {
4001     \bool_if:NTF \l__enumext_minipage_active_v_bool
4002       {
4003         \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
4004           {
4005             \msg_warning:nn { enumext } { missing-miniright }
4006             \miniright
4007           }
4008         \int_gzero:N \g__enumext_minipage_stat_int
4009         \__enumext_unskip_unkern: % remove \topsep + [\partopsep]
4010         \end__enumext_mini_page
4011         \par\addvspace{ \l__enumext_minipage_after_skip }
4012       }
4013       {
4014         \__enumext_keyans_multicols_stop:
4015       }
4016     \bool_set_false:N \l__enumext_keyans_env_bool
4017     \__enumext_after_stop_list_v:
4018     \__enumext_vspace_below_v:
4019   }
```

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

The function __enumext_keyans_set_item_width: will set the value of \itemwidth taking into account the value established by the list-offset key.

```
4020 \cs_new_protected:Nn \__enumext_keyans_set_item_width:
4021   {
4022     \dim_set:Nn \itemwidth { \linewidth }
4023     \dim_compare:nT
4024       {
4025         \l__enumext_listoffset_v_dim != \c_zero_dim
4026       }
4027       {
```

```
4028              \dim_sub:Nn \itemwidth { \l__enumext_listoffset_v_dim }
4029          }
4030      }
```

(*End of definition for* __enumext_keyans_set_item_width:.)

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

```
4031  \NewDocumentEnvironment{keyans}{ O{} }
4032    {
4033      \__enumext_keyans_safe_exec:
4034      \__enumext_keyans_parse_keys:n {#1}
4035      \__enumext_before_list_v:
4036      \__enumext_start_list:nn
4037        { \tl_use:N \l__enumext_label_v_tl }
4038        {
4039          \__enumext_list_arg_two_v:
4040          \__enumext_before_keys_exec_v:
4041        }
4042      \__enumext_keyans_set_item_width:
4043      \__enumext_after_args_exec_v:
4044    }
4045    {
4046      \__enumext_check_starred_cmd:n { item }
4047      \__enumext_second_part_v:
4048    }
```

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

13.41   Tagging PDF support for non-standart list environments

The LaTeX release 2022-06-01 brings automatic support for *tagged* PDF in several aspects, including the standard *list environments* and the list environment. Unfortunately non-standard *list environments* like keyanspic or the horizontal list environments enumext* and keyans* are not structured in a nice way, i.e. the expected result in the PDF file is the expected one, but the underlying structure is not correct. In simple terms, for *tagged* PDF a list environment is a list environment, no matter what it looks like in the PDF file.

To maintain a correct list structure when \DocumentMetadata is active, it is necessary to do some things manually using tagpdf[17] and ltsockets[19]. This implementation is an adaptation of my answer thanks to Ulrike Fischer's comments in How can I modify my \item redefinition to be compatible with tagging-pdf.

13.41.1   Socket for tagging support in enumext* and keyans*

start-list-tags   We will first define the necessary sockets and their behavior for enumext* and keyans*.
stop-start-tags
stop-list-tags
__enumext_start_list_tag:n
__enumext_stop_start_list_tag:
__enumext_stop_list_tag:n

```
4049  \socket_new:nn {tagsupport/__enumext/starred}{ 1 }
4050  \socket_new_plug:nnn {tagsupport/__enumext/starred} {start-list-tags}
4051    {
4052      \tag_resume:n {#1}
4053      \tag_mc_end_push:
4054        \tag_struct_begin:n {tag=LI}
4055          \tag_struct_begin:n {tag=Lbl}
4056            \tag_mc_begin:n {tag=Lbl}
4057    }
4058  \socket_new_plug:nnn {tagsupport/__enumext/starred} {stop-start-tags}
4059    {
4060            \tag_mc_end:
4061          \tag_struct_end:n {tag=Lbl}
4062          \tag_struct_begin:n {tag=LBody}
4063            \tag_struct_begin:n {tag=text-unit}
4064              \tag_struct_begin:n {tag=text}
4065    }
4066  \socket_new_plug:nnn {tagsupport/__enumext/starred} {stop-list-tags}
4067    {
4068              \tag_struct_end:n {tag=text}
4069            \tag_struct_end:n {tag=text-unit}
4070          \tag_struct_end:n {tag=LBody}
4071        \tag_struct_end:n {tag=LI}
4072      \tag_mc_begin_pop:n {}
4073      \tag_suspend:n {#1}
4074    }
```

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

```
4075  \cs_new_protected_nopar:Npn \__enumext_start_list_tag:n #1
4076    {
```

```
4077        \IfDocumentMetadataTF
4078          {
4079            \socket_assign_plug:nn {tagsupport/__enumext/starred} {start-list-tags}
4080            \socket_use:nn {tagsupport/__enumext/starred} {#1}
4081          } {}
4082      }
4083  \cs_new_protected_nopar:Nn \__enumext_stop_start_list_tag:
4084      {
4085        \IfDocumentMetadataTF
4086          {
4087            \socket_assign_plug:nn {tagsupport/__enumext/starred} {stop-start-tags}
4088            \socket_use:nn {tagsupport/__enumext/starred} { }
4089          } {}
4090      }
4091  \cs_new_protected_nopar:Npn \__enumext_stop_list_tag:n #1
4092      {
4093        \IfDocumentMetadataTF
4094          {
4095            \socket_assign_plug:nn {tagsupport/__enumext/starred} {stop-list-tags}
4096            \socket_use:nn {tagsupport/__enumext/starred} {#1}
4097          } {}
4098      }
```

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

13.41.2   Socket for tagging support in keyanspic

start-list-tags
stop-start-tags
stop-list-tags
__enumext_anspic_start_list_tag:
__enumext_anspic_stop_start_list_tag:
__enumext_anspic_stop_list_tag:

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

```
4099  \socket_new:nn {tagsupport/__enumext/keyanspic}{ 0 }
4100  \socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {start-list-tags}
4101      {
4102        \tag_resume:n {keyanspic}
4103        \tag_mc_end_push:
4104          \tag_struct_begin:n {tag=LI}
4105            \tag_struct_begin:n {tag=Lbl}
4106              \tag_mc_begin:n {tag=Lbl}
4107      }
4108  \socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {stop-start-tags}
4109      {
4110              \tag_mc_end:
4111            \tag_struct_end:n {tag=Lbl}
4112          \tag_struct_begin:n {tag=LBody}
4113            \tag_struct_begin:n {tag=text-unit}
4114              \tag_struct_begin:n {tag=text}
4115                \tag_mc_begin:n {tag=text}
4116      }
4117  \socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {stop-list-tags}
4118      {
4119                \tag_mc_end:
4120              \tag_struct_end:n {tag=text}
4121            \tag_struct_end:n {tag=text-unit}
4122          \tag_struct_end:n {tag=LBody}
4123        \tag_struct_end:n {tag=LI}
4124      \tag_mc_begin_pop:n {}
4125      \tag_suspend:n {keyanspic}
4126      }
```

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

```
4127  \cs_new_protected_nopar:Nn \__enumext_anspic_start_list_tag:
4128      {
4129        \IfDocumentMetadataTF
4130          {
4131            \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {start-list-tags}
4132            \socket_use:n {tagsupport/__enumext/keyanspic}
4133          } {}
4134      }
4135  \cs_new_protected_nopar:Nn \__enumext_anspic_stop_start_list_tag:
4136      {
4137        \IfDocumentMetadataTF
4138          {
4139            \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {stop-start-tags}
4140            \socket_use:n {tagsupport/__enumext/keyanspic}
4141          } {}
```

```
4142        }
4143 \cs_new_protected_nopar:Nn \__enumext_anspic_stop_list_tag:
4144    {
4145      \IfDocumentMetadataTF
4146        {
4147          \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {stop-list-tags}
4148          \socket_use:n {tagsupport/__enumext/keyanspic}
4149        } {}
4150    }
```

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

13.42 The environment keyanspic and \anspic

The `keyanspic` environment is a `list` based environment that uses the same configuration for *"spacing"* and ⟨*label*⟩ as the `keyans` environment, but it does not use \item. The ⟨*contents*⟩ are passed to the environment by means of the \anspic command as replacement for \item command and placed inside `minipage` environments, with the ⟨*label*⟩ centered *"above"* or *"below"*, adjusting *widths* and *position* according to the options passed to the environment.
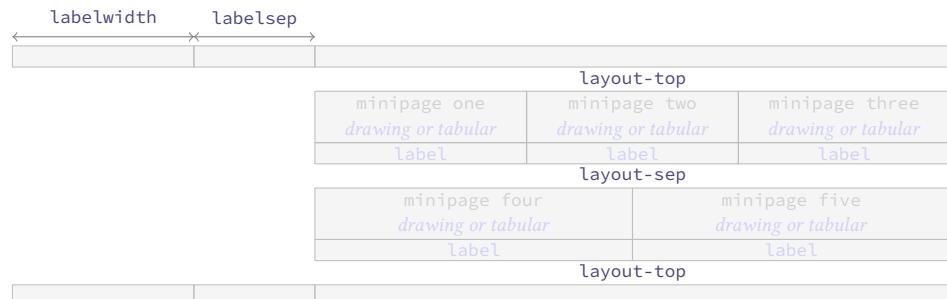
Figure 12: Representation of the `keyanspic` spacing in enumext.

In order for the `keyanspic` environment and the \anspic command to work correctly, we need to set and export some variables in the first part of the environment definition and pass them to \anspic which is executed in the second part of the environment. This implementation is adapted from the answer given by Enrico Gregorio (@egreg) in How to process the body of an environment and divide it by a \macro?.

13.42.1 The environment keyanspic

label-pos
label-sep
layout-sty
layout-sep
layout-top
mark-ans
mark-pos
mark-sep
save-sep
wrap-opt
wrap-ans*
show-ans
show-pos

First we define the key that allows us to process the position of the ⟨*label*⟩ centered "above" or "below" which will be `label-pos`, the vertical separation of these from *drawing or tabular* will be handled with the key `label-sep`. The *"layout style"* will be handled with the key `layout-sty` will take two values separated by comma {⟨*n° upper, n° lower*⟩} and will determine the number of `minipage` environments in which all arguments of \anspic will be printed at the "upper" and "lower" within the environments separated by the value of the key `layout-sep`. The vertical space "top" and "bottom" of the environment will be handled with the key `layout-top`.

```
4151 \keys_define:nn { enumext / keyanspic }
4152    {
4153      label-pos .choice:,
4154      label-pos / above   .code:n =
4155                            \bool_set_true:N \l__enumext_anspic_label_above_bool
4156                            \str_set:Nn \l__enumext_anspic_mini_pos_str { t },
4157      label-pos / below   .code:n =
4158                            \bool_set_false:N \l__enumext_anspic_label_above_bool
4159                            \str_set:Nn \l__enumext_anspic_mini_pos_str { b },
4160      label-pos / unknown .code:n =
4161                            \msg_error:nneee { enumext } { unknown-choice }
4162                             { label-pos } { above,~ below } { \exp_not:n {#1} },
4163      label-pos  .initial:n      = below,
4164      label-pos  .value_required:n = true,
4165      label-sep  .skip_set:N       = \l__enumext_anspic_label_sep_skip,
4166      label-sep  .value_required:n = true,
4167      layout-sty .tl_set:N         = \l__enumext_anspic_layout_style_tl,
4168      layout-sty .value_required:n = true,
4169      layout-sep .code:n           = \keys_set:nn { enumext / keyans } { parsep = #1 },
4170      layout-sep .value_required:n = true,
4171      layout-top .code:n           = \keys_set:nn { enumext / keyans } { topsep = #1 },
4172      layout-top .value_required:n = true,
4173      mark-ans   .code:n           = \keys_set:nn { enumext / keyans } { mark-ans = #1 },
4174      mark-ans   .value_required:n = true,
4175      mark-pos   .code:n           = \keys_set:nn { enumext / keyans } { mark-pos = #1 },
4176      mark-pos   .value_required:n = true,
```

```
4177    mark-sep    .code:n            = \keys_set:nn { enumext / keyans } { mark-sep = #1 },
4178    mark-sep    .value_required:n = true,
4179    save-sep    .code:n            = \keys_set:nn { enumext / keyans } { save-sep = #1 },
4180    save-sep    .value_required:n = true,
4181    wrap-opt    .code:n            = \keys_set:nn { enumext / keyans } { wrap-opt = #1 },
4182    wrap-opt    .value_required:n = true,
4183    wrap-ans*   .code:n            = \keys_set:nn { enumext / keyans } { wrap-ans* = #1 },
4184    wrap-ans*   .value_required:n = true,
4185    show-ans    .code:n            = \keys_set:nn { enumext / keyans } { show-ans = #1 },
4186    show-ans    .value_required:n = true,
4187    show-pos    .code:n            = \keys_set:nn { enumext / keyans } { show-pos = #1 },
4188    show-pos    .value_required:n = true,
4189    unknown     .code:n            = {
4190                                       \tl_set:Nn \l__enumext_envir_name_tl { keyanspic }
4191                                       \__enumext_keyans_unknown_keys:n {#1}
4192                                     },
4193    }
```

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

__enumext_keyans_pic_safe_exec:
__enumext_keyans_pic_parse_keys:n
__enumext_keyans_pic_skip_abs:N
__enumext_keyans_pic_arg_two:

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

```
4194 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
4195   {
4196     \int_incr:N \l__enumext_keyans_pic_level_int
4197     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
4198       {
4199         \msg_error:nn { enumext } { keyanspic-nested }
4200       }
4201     \__enumext_keyans_name_and_start:
4202   }
```

Parse $[\langle key = val \rangle]$ for keyanspic environment.

```
4203 \cs_new_protected:Npn \__enumext_keyans_pic_parse_keys:n #1
4204   {
4205     \tl_if_novalue:nF {#1}
4206       {
4207         \keys_set:nn { enumext / keyanspic } {#1}
4208       }
4209   }
```

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

```
4210 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
4211   {
4212     \dim_compare:nNnT { #1 } < { \c_zero_dim }
4213       {
4214         \skip_set:Nn #1 { -#1 }
4215       }
4216   }
```

The __enumext_keyans_pic_arg_two: function will be used in the *second argument* of the list environment that defines the keyanspic environment, with this we will take the configuration of the *"spaces"* and the keys label, wrap-label, parsep and topsep from the keyans environment. The first thing we need to do is set the boolean variable \l__enumext_leftmargin_tmp_v_bool handled by the list-indent key to "false", then copy the definition of the second list argument from the keyans environment definition and make sure that \parsep does not have a negative value.

```
4217 \cs_new_protected:Npn \__enumext_keyans_pic_arg_two:
4218   {
4219     \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
4220     \__enumext_list_arg_two_v:
4221     \__enumext_keyans_pic_skip_abs:N \parsep
```

Now we increment the counter enumXv of the keyans environment and save the *total height* of the $\langle label \rangle$ in \l__enumext_anspic_label_htdp_dim used by \anspic and we will adjust the values of \parsep only if the key label-pos is set to *below*.

```
4222     \bool_if:NF \l__enumext_anspic_label_above_bool
4223       {
4224         \stepcounter { enumXv }
4225         \hbox_set:Nn \l__enumext_anspic_label_box { \l__enumext_label_v_tl }
4226         \dim_set:Nn \l__enumext_anspic_label_htdp_dim
4227           {
```

```
4228                \box_ht_plus_dp:N \l__enumext_anspic_label_box
4229              }
4230          \skip_add:Nn \parsep
4231            {
4232              \l__enumext_anspic_label_htdp_dim
4233              + \box_dp:N \strutbox
4234              + \l__enumext_anspic_label_sep_skip
4235            }
4236        }
```

Finally we *adjust* the value of \leftmargin and \topsep then set \listparindent, \partopsep and \itemsep to zero so that the *horizontal* and *vertical* space is not affected.

```
4237        \dim_add:Nn \leftmargin { -\l__enumext_labelwidth_v_dim - \l__enumext_labelsep_v_dim }
4238        \ignorespaces
4239        \skip_add:Nn \topsep { 0.5\box_dp:N \strutbox }
4240        \dim_zero:N  \listparindent
4241        \skip_zero:N \partopsep
4242        \skip_zero:N \itemsep
4243    }
```

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

keyanspic   Now we define the environment keyanspic. For compatibility with *tagged* PDF we must use the \begin{list} form and a lot of conditional code using \IfDocumentMetadataTF. We will first stop the code for automatic *tagged* PDF for list environments, redefine \item so that it cannot be used, and stop the code for automatic *tagged* PDF for the keyanspic environment.

```
4244  \NewDocumentEnvironment{keyanspic}{ o }
4245    {
4246      \__enumext_keyans_pic_safe_exec:
4247      \__enumext_keyans_pic_parse_keys:n {#1}
4248      \begin{list} { } { \__enumext_keyans_pic_arg_two: }
4249      \IfDocumentMetadataTF
4250        {
4251          \tag_suspend:n {list}
4252        }{}
4253      \item[] \scan_stop:
4254      \RenewDocumentCommand \item {}
4255        {
4256          \msg_error:nn { enumext } { keyanspic-item-cmd }
4257        }
4258      \IfDocumentMetadataTF
4259        {
4260          \tag_resume:n {keyanspic}
4261          \tag_tool:n {para/tagging=false}
4262          \tag_suspend:n {keyanspic}
4263        } { }
4264    }
```

In the second part of the environment definition we will manually place our code for *tagged* PDF and execute the command \anspic using the __enumext_anspic_exec: function.

```
4265      {
4266        \IfDocumentMetadataTF
4267          {
4268            \tag_resume:n {keyanspic}
4269            \tag_mc_end_push:
4270            \tag_struct_begin:n {tag=L,attribute=enumerate}
4271          } { }
4272        \__enumext_anspic_exec:
4273        \IfDocumentMetadataTF
4274          {
4275            \tag_suspend:n {keyanspic}
4276          } { }
4277        \end{list}
4278        \IfDocumentMetadataTF
4279          {
4280            \tag_struct_end:n {tag=L}
4281            \tag_mc_begin_pop:n {}
4282            \tag_struct_end:n {tag=L}
4283            \tag_mc_begin_pop:n {}
4284          } { }
```

Finally we check if `\anspic*` has been used, set the counter enumXvi to zero and apply our "adjusted" vertical space bottom.

```
4285        \__enumext_check_starred_cmd:n { anspic }
4286        \setcounter { enumXvi } { 0 }
4287        \bool_if:NTF \l__enumext_anspic_label_above_bool
4288          {
4289            \par\addvspace{ 0.5\box_dp:N \strutbox }
4290          }
4291          {
4292            \par
4293            \addvspace
4294              {
4295                \dim_eval:n
4296                  {
4297                    \l__enumext_anspic_label_htdp_dim + \box_ht_plus_dp:N \strutbox
4298                    + \l__enumext_anspic_label_sep_skip + \l__enumext_topsep_v_skip
4299                  }
4300              }
4301          }
4302        }
```

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

13.42.2   The command `\anspic`

The `\anspic` command take three arguments, the *starred versions* `\anspic*`[⟨*content*⟩] *store* the current ⟨*label*⟩ next to the *optional argument* [⟨*content*⟩] in the *sequence* and *prop list* defined by `save-ans` key. The third *mandatory argument* {⟨*drawing or tabular*⟩} is NOT *stored* in the *sequence* or *prop list*.

🔵 One of the complications here to make the `keyanspic` environment compatible with *tagged* PDF is the position of ⟨*label*⟩, the `\anspic` command processes the arguments in order, where `#1` and `#2` correspond to ⟨*label*⟩ and `#3` to the mandatory argument and puts all this inside a `minipage` environment. If `#1` and `#2`, that is ⟨*label*⟩, is above `#3` there are no problems with *tagged* PDF, but if `#3` comes first the list created with *tagged* PDF will not be correct.

`\anspic`
`\__enumext_anspic_body_dim:n`
`\__enumext_anspic_label:nn`
`\__enumext_anspic_label_pos:nnn`
`\__enumext_anspic_args:nnn`
`\__enumext_anspic_print:n`
`\__enumext_anspic_print:e`
`\__enumext_anspic_print:V`
`\__enumext_anspic_row:n`
`__enumext_anspic_exec:`

We check that the command is active in the `keyanspic` environment only if the `save-ans` key is present, otherwise we return an error. The three arguments are handled by the function `\__enumext_anspic_args:nnn` and stored in the sequence `\l__enumext_anspic_args_seq` which is processed by the `keyanspic` environment.

```
4303  \NewDocumentCommand \anspic { s o +m }
4304    {
4305      \bool_if:NF \l__enumext_store_active_bool
4306        {
4307          \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
4308        }
4309      \int_compare:nNnT { \l__enumext_level_int } > { 1 }
4310        {
4311          \msg_error:nn { enumext } { keyanspic-wrong-level }
4312        }
4313      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
4314        {
4315          \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
4316        }
4317      \seq_put_right:Nn \l__enumext_anspic_args_seq
4318        {
4319          \__enumext_anspic_args:nnn { #1 } { #2 } { #3 }
4320        }
4321    }
```

The `\__enumext_anspic_body_dim:n` function will set the value of `\l__enumext_anspic_body_htdp_-dim` equal to the "height plus depth" of the *mandatory argument* if the key `label-pos` is set "below".

```
4322  \cs_new_protected:Npn \__enumext_anspic_body_dim:n #1
4323    {
4324      \bool_if:NF \l__enumext_anspic_label_above_bool
4325        {
4326          \IfDocumentMetadataTF
4327            {
4328              \tag_suspend:n {keyanspic}
4329            } { }
4330          \vbox_set:Nn \l__enumext_anspic_body_box { #1 }
4331          \dim_set:Nn \l__enumext_anspic_body_htdp_dim
4332            {
4333              \box_ht_plus_dp:N \l__enumext_anspic_body_box
```

```
4334                    }
4335            \IfDocumentMetadataTF
4336                {
4337                    \tag_resume:n {keyanspic}
4338                } { }
4339            }
4340        }
```

The `\__enumext_anspic_label:nn` function will process inside `\makebox` the *starred argument* '`*`' and *optional argument* passed to the command. Here we will store the ⟨*label*⟩ and *optional argument* in *prop list* and *sequence* and execute the `show-ans`, `show-pos`, `font`, `wrap-label`, `wrap-ans*` and `wrap-opt` keys.

```
4341  \cs_new_protected:Npn \__enumext_anspic_label:nn #1 #2
4342    {
4343      \makebox[ \l__enumext_anspic_mini_width_dim ][ c ]
4344        {
4345          \bool_if:nTF { #1 }
4346            {
4347              \bool_set_true:N \l__enumext_item_wrap_key_bool
4348              \bool_set_true:N \l__enumext_wrap_label_v_bool
4349              \__enumext_keyans_save_item_opt:n { #2 }
4350              \__enumext_keyans_addto_prop:n { #2 }
4351              \__enumext_keyans_store_ref:
4352              \__enumext_keyans_addto_seq:n { #2 }
4353              \int_gincr:N \g__enumext_check_starred_cmd_int
4354              \__enumext_keyans_show_ans:
4355              \__enumext_keyans_show_pos:
4356              \makebox[ \l__enumext_labelwidth_v_dim ][c]
4357                {
4358                  \tl_use:N \l__enumext_label_font_style_v_tl
4359                  \__enumext_keyans_wrapper_label:n { \l__enumext_label_vi_tl }
4360                }
4361              \skip_horizontal:n { \l__enumext_labelsep_v_dim }
4362              \__enumext_keyans_show_item_opt:
4363            }
4364            {
4365              \bool_set_false:N \l__enumext_item_wrap_key_bool
4366              \tl_use:N \l__enumext_label_font_style_v_tl
4367              \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl }
4368            }
4369        }
4370    }
```

The function `\__enumext_anspic_label_pos:nnn` will be in charge of handling the *"counter"* and the position of the ⟨*label*⟩, set by `label-pos` key which will have the same configuration as the `keyans` environment.

```
4371  \cs_new_protected:Npn \__enumext_anspic_label_pos:nnn #1 #2 #3
4372    {
4373      \stepcounter { enumXvi }
4374      \__enumext_anspic_body_dim:n { #3 }
4375      \bool_if:NTF \l__enumext_anspic_label_above_bool
4376        {
4377          \__enumext_anspic_label:nn { #1 } { #2 }
4378        }
4379        {
4380          \raisebox
4381            {
4382              -\dim_eval:n
4383                {
4384                  \l__enumext_anspic_label_htdp_dim
4385                  + \l__enumext_anspic_body_htdp_dim
4386                  + \box_dp:N \strutbox
4387                  + \l__enumext_anspic_label_sep_skip
4388                }
4389            }
4390            [ 0pt ] [ 0pt ]
4391            {
4392              \__enumext_anspic_label:nn { #1 } { #2 }
4393            }
4394        }
4395    }
4396  %
```

The __enumext_anspic_args:nnn function will be responsible for placing the code compatible with *tagged* PDF and the arguments within the \l__enumext_anspic_args_seq sequence which will be processed by the __enumext_anspic_print:n function in the second part of the definition of the keyanspic environment.

```
4397 \cs_new_protected:Nn \__enumext_anspic_args:nnn
4398   {
4399     \__enumext_anspic_start_list_tag:
4400     \__enumext_anspic_label_pos:nnn { #1 } { #2 } { #3 }
4401     \__enumext_anspic_stop_start_list_tag:
4402     \bool_if:NTF \l__enumext_anspic_label_above_bool
4403       {
4404         \\[\l__enumext_anspic_label_sep_skip] #3
4405       }
4406       {
4407         \\ #3
4408       }
4409     \__enumext_anspic_stop_list_tag:
4410   }
```

The value {⟨*n° upper, n° lower*⟩} passed to the layout-sty key is split by comma and is handled directly by the function __enumext_anspic_print:n and passed to the function __enumext_anspic_row:n.

```
4411 \cs_new_protected:Nn \__enumext_anspic_print:n
4412   {
4413     \clist_map_function:nN { #1 } \__enumext_anspic_row:n
4414   }
4415 \cs_generate_variant:Nn \__enumext_anspic_print:n { e, V }
```

The function __enumext_anspic_row:n will set the *widths* for the minipage environments and place *all arguments* passed to \anspic *saved* in the \l__enumext_anspic_args_seq sequence inside them.

```
4416 \cs_new_protected:Nn \__enumext_anspic_row:n
4417   {
4418     \dim_set:Nn \l__enumext_anspic_mini_width_dim { \linewidth / #1 }
4419     \int_set:Nn \l__enumext_anspic_above_int { \l__enumext_anspic_below_int }
4420     \int_set:Nn \l__enumext_anspic_below_int { \l__enumext_anspic_above_int + #1 }
4421     \int_step_inline:nnn
4422       { \l__enumext_anspic_above_int + 1 }
4423       { \l__enumext_anspic_below_int }
4424       {
4425         \IfDocumentMetadataTF
4426           {
4427             \tag_suspend:n {minipage}
4428           } { }
4429         \begin{minipage}[ \l__enumext_anspic_mini_pos_str ]{ \l__enumext_anspic_mini_width_dim }
4430           \centering
4431           \seq_item:Nn \l__enumext_anspic_args_seq { ##1 }
4432         \end{minipage}
4433         \IfDocumentMetadataTF
4434           {
4435             \tag_resume:n {minipage}
4436           } { }
4437       }
4438     \par
4439   }
```

The __enumext_anspic_exec: function will execute all the code in the \anspic command in the second argument of the keyanspic environment definition. If the key layout-sty is not set, everything will be printed on a *single line*.

```
4440 \cs_new_protected:Nn \__enumext_anspic_exec:
4441   {
4442     \tl_if_empty:NTF \l__enumext_anspic_layout_style_tl
4443       {
4444         \__enumext_anspic_print:e { \seq_count:N \l__enumext_anspic_args_seq }
4445       }
4446       {
4447         \__enumext_anspic_print:V \l__enumext_anspic_layout_style_tl
4448       }
4449   }
```

(*End of definition for* \anspic *and others. This function is documented on page 17.*)

13.43 The horizontal environments

Generating *horizontal list environments* is NOT as simple as standard LATEX list environments. The fundamental part of the code is adapted from the `shortlst` package to a more modern version using `expl3`. It is not possible to redefine `\item` and `\makelabel` using `\RenewDocumentCommand` as in the vertical *non starred* versions.

To achieve the *horizontal list environments* we will capture the `\item` command and the ⟨*content*⟩ of this in *horizontal box* using `\makebox` for the `label` and a `minipage` environment for the ⟨*content*⟩ passed to `\item`, we will also add the *optional argument* (⟨*number*⟩) to `\item` to be able to *join columns* horizontally, in simple terms, we want `\item` to behave in the same way as in the `enumext` environment but adding an *first optional argument* (⟨*number*⟩).

A side effect is the limitation of using `\item` in this way *without* using `\RenewDocumentCommand`, which loses the original definition and affects the *standard list environments* provided by LATEX and any environment defined using base `list` environment, including: `itemize`, `enumerate`, `description`, `quote`, `quotation`, `verse`, `center`, `flushleft`, `flushright`, `verbatim`, `tabbing`, `trivlist`, `list` and all environments created with `\newtheorem`.

💣 One way to get around this is to use something like:

`\AddToHook{env/enumerate/before}{recover original \item definition}`

inside `minipage`, but in my partial tests this does not have the desired effect and the vertical and horizontal spacing is distorted. For now this will remain as a limitation and I will see if it is feasible to implement it in the future.

🔷 For compatibility with the *tagged* PDF we close the environments according to the presence or not of the `mini-env` key.

13.43.1 Functions for item box width

`\__enumext_starred_columns_set_vii:`
`\__enumext_starred_columns_set_viii:`

We set the default value for the *width of the box* containing the ⟨*content*⟩ of the items for `enumext*` environment.

```
4450 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
4451   {
4452     \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
4453       {
4454         \dim_set:Nn \l__enumext_columns_sep_vii_dim
4455           {
4456             ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
4457             / \l__enumext_columns_vii_int
4458           }
4459       }
4460     \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - 1 }
4461     \dim_set:Nn \l__enumext_item_width_vii_dim
4462       {
4463         ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
4464         / \l__enumext_columns_vii_int
4465         - \l__enumext_labelwidth_vii_dim
4466         - \l__enumext_labelsep_vii_dim
4467       }
```

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

```
4468     \dim_compare:nNnT { \l__enumext_rightmargin_vii_dim } > { \c_zero_dim }
4469       {
4470         \dim_sub:Nn \l__enumext_item_width_vii_dim
4471           {
4472             ( \l__enumext_rightmargin_vii_dim * \l__enumext_tmpa_vii_int )
4473             / \l__enumext_columns_vii_int
4474           }
4475         \dim_add:Nn \l__enumext_columns_sep_vii_dim
4476           {
4477             \l__enumext_rightmargin_vii_dim
4478           }
4479       }
4480   }
```

Same implementation for the `keyans*` environment.

```
4481 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
4482   {
4483     \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
4484       {
4485         \dim_set:Nn \l__enumext_columns_sep_viii_dim
4486           {
4487             ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
4488             / \l__enumext_columns_viii_int
4489           }
4490       }
```

```
4491      \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - 1 }
4492      \dim_set:Nn \l__enumext_item_width_viii_dim
4493        {
4494          ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
4495          / \l__enumext_columns_viii_int
4496          - \l__enumext_labelwidth_viii_dim
4497          - \l__enumext_labelsep_viii_dim
4498        }
4499      \dim_compare:nNnT { \l__enumext_rightmargin_viii_dim } > { \c_zero_dim }
4500        {
4501          \dim_sub:Nn \l__enumext_item_width_viii_dim
4502            {
4503              ( \l__enumext_rightmargin_viii_dim * \l__enumext_tmpa_vii_int )
4504              / \l__enumext_columns_viii_int
4505            }
4506          \dim_add:Nn \l__enumext_columns_sep_viii_dim
4507            {
4508              \l__enumext_rightmargin_viii_dim
4509            }
4510        }
4511    }
```

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

13.43.2   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.

```
4512  \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
4513    {
4514      \int_set:Nn \l__enumext_joined_item_vii_int {#1}
4515      \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
4516        {
4517          \msg_warning:nnee { enumext } { item-joined }
4518            { \int_use:N \l__enumext_joined_item_vii_int }
4519            { \int_use:N \l__enumext_columns_vii_int }
4520          \int_set:Nn \l__enumext_joined_item_vii_int
4521            {
4522              \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4523            }
4524        }
4525      \int_compare:nNnT
4526        { \l__enumext_joined_item_vii_int }
4527        >
4528        { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4529        {
4530          \msg_warning:nnee { enumext } { item-joined-columns }
4531            { \int_use:N \l__enumext_joined_item_vii_int }
4532            {
4533              \int_eval:n
4534                { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4535            }
4536          \int_set:Nn \l__enumext_joined_item_vii_int
4537            {
4538              \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4539            }
4540        }
4541      \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { 1 }
4542        {
4543          \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
4544          \int_decr:N \l__enumext_joined_item_aux_vii_int
4545          \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
4546          \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
4547          \dim_set:Nn \l__enumext_joined_width_vii_dim
4548            {
4549              \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
4550              + (  \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
4551                + \l__enumext_columns_sep_vii_dim
4552              )*\l__enumext_joined_item_aux_vii_int
4553            }
4554          \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
```

```
4555          }
4556          {
4557              \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
4558              \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
4559          }
4560      }
```

Same implementation for the keyans* environment.

```
4561  \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
4562      {
4563          \int_set:Nn \l__enumext_joined_item_viii_int {#1}
4564          \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
4565              {
4566                  \msg_warning:nnee { enumext } { item-joined }
4567                      { \int_use:N \l__enumext_joined_item_viii_int }
4568                      { \int_use:N \l__enumext_columns_viii_int }
4569                  \int_set:Nn \l__enumext_joined_item_viii_int
4570                      {
4571                          \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4572                      }
4573              }
4574          \int_compare:nNnT
4575              { \l__enumext_joined_item_viii_int }
4576                  >
4577              { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4578              {
4579                  \msg_warning:nnee { enumext } { item-joined-columns }
4580                      { \int_use:N \l__enumext_joined_item_viii_int }
4581                      {
4582                          \int_eval:n
4583                              { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4584                      }
4585                  \int_set:Nn \l__enumext_joined_item_viii_int
4586                      {
4587                          \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4588                      }
4589              }
4590          \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { 1 }
4591              {
4592                  \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
4593                  \int_decr:N \l__enumext_joined_item_aux_viii_int
4594                  \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
4595                  \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
4596                  \dim_set:Nn \l__enumext_joined_width_viii_dim
4597                      {
4598                          \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
4599                          + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
4600                              + \l__enumext_columns_sep_viii_dim
4601                          )*\l__enumext_joined_item_aux_viii_int
4602                      }
4603                  \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
4604              }
4605              {
4606                  \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
4607                  \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
4608              }
4609      }
```

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

13.43.3   Functions for mini-env, mini-right and mini-right* keys

__enumext_start_mini_vii:
__enumext_stop_mini_vii:

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

```
4610  \cs_new_protected:Nn \__enumext_start_mini_vii:
4611      {
4612          \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
4613              {
4614                  \dim_set:Nn \l__enumext_minipage_left_vii_dim
4615                      {
```

```
4616              \linewidth
4617                - \l__enumext_minipage_right_vii_dim
4618                - \l__enumext_minipage_hsep_vii_dim
4619            }
4620          \bool_set_true:N \l__enumext_minipage_active_vii_bool
4621          \dim_gset_eq:NN
4622            \g__enumext_minipage_right_vii_dim
4623            \l__enumext_minipage_right_vii_dim
4624          \__enumext_mini_addvspace_vii:
4625          \nointerlineskip\noindent
4626          \__enumext_mini_page{ \l__enumext_minipage_left_vii_dim }
4627        }
4628      }
```

The function `\__enumext_stop_mini_vii:` closes the `__enumext_mini_page` environment on the *"left side"*, applies `\hfill` and set the variable `\g__enumext_minipage_active_vii_bool` to *"true"* which will be used in the function `\__enumext_after_env:nn` to execute the `minipage` on the *"right side"*. At this point we will execute the `\__enumext_stop_list:` and `\__enumext_stop_store_level_vii:` functions stopping the `list` environment and the level saving mechanism for storage in *sequence* of the `\anskey` command and `anskey*` environment. This function is passed to the `\__enumext_after_list_vii:` function in the second part of the `enumext*` environment definition (§13.44).

```
4629 \cs_new_protected:Nn \__enumext_stop_mini_vii:
4630   {
4631     \bool_if:NTF \l__enumext_minipage_active_vii_bool
4632       {
4633         \__enumext_stop_list:
4634         \__enumext_stop_store_level_vii:
4635         \IfDocumentMetadataTF { \tag_resume:n {enumext*} } { }
4636         \end__enumext_mini_page
4637         \hfill
4638         \bool_gset_true:N \g__enumext_minipage_active_vii_bool
4639       }
4640       {
4641         \__enumext_stop_list:
4642         \__enumext_stop_store_level_vii:
4643       }
4644   }
```

(*End of definition for* `\__enumext_start_mini_vii:` *and* `\__enumext_stop_mini_vii:`.)

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

```
4645 \__enumext_after_env:nn {enumext*}
4646   {
4647     \bool_if:NT \g__enumext_minipage_active_vii_bool
4648       {
4649         \__enumext_minipage:w [ t ] { \g__enumext_minipage_right_vii_dim }
4650           \legacy_if_gset_false:n { @minipage }
4651           \skip_vertical:N \c_zero_skip
4652           \par\addvspace { \g__enumext_minipage_right_skip }
4653           \bool_if:NF \g__enumext_minipage_center_vii_bool
4654             {
4655               \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
4656                 {
4657                   \centering
4658                 }
4659             }
4660           \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
4661             {
4662               \tl_use:N \g__enumext_miniright_code_vii_tl
4663             }
4664           \box_use_drop:N \l__enumext_miniright_code_vii_box
4665           \skip_vertical:N \c_zero_skip
4666         \__enumext_endminipage:
4667         \par\addvspace{ \g__enumext_minipage_after_skip }
4668       }
4669     \bool_gset_false:N \g__enumext_minipage_active_vii_bool
4670     \bool_gset_true:N \g__enumext_minipage_center_vii_bool
4671     \tl_gclear:N \g__enumext_miniright_code_vii_tl
4672     \dim_gzero:N \g__enumext_minipage_right_vii_dim
```

```
4673        \bool_gset_false:N \g__enumext_starred_bool
4674      }
```

__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.

```
4675  \cs_new_protected:Nn \__enumext_start_mini_viii:
4676    {
4677      \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
4678        {
4679          \dim_set:Nn \l__enumext_minipage_left_viii_dim
4680            {
4681              \linewidth
4682              - \l__enumext_minipage_right_viii_dim
4683              - \l__enumext_minipage_hsep_viii_dim
4684            }
4685          \bool_set_true:N \l__enumext_minipage_active_viii_bool
4686          \dim_gset_eq:NN
4687            \g__enumext_minipage_right_viii_dim
4688            \l__enumext_minipage_right_viii_dim
4689          \__enumext_mini_addvspace_viii:
4690          \nointerlineskip\noindent
4691          \__enumext_mini_page{ \l__enumext_minipage_left_viii_dim }
4692        }
4693    }
4694  \cs_new_protected:Nn \__enumext_stop_mini_viii:
4695    {
4696      \bool_if:NTF \l__enumext_minipage_active_viii_bool
4697        {
4698          \__enumext_stop_list:
4699          \IfDocumentMetadataTF { \tag_resume:n {keyans*} } { }
4700          \end__enumext_mini_page
4701          \hfill
4702          \bool_gset_true:N \g__enumext_minipage_active_viii_bool
4703        }
4704        {
4705          \__enumext_stop_list:
4706        }
4707    }
4708  \__enumext_after_env:nn {keyans*}
4709    {
4710      \bool_if:NT \g__enumext_minipage_active_viii_bool
4711        {
4712          \__enumext_mini_page{ \g__enumext_minipage_right_viii_dim }
4713            \par\addvspace { \g__enumext_minipage_right_skip }
4714            \bool_if:NF \g__enumext_minipage_center_viii_bool
4715              {
4716                \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
4717                  {
4718                    \centering
4719                  }
4720              }
4721            \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
4722              {
4723                \tl_use:N \g__enumext_miniright_code_viii_tl
4724              }
4725            \box_use_drop:N \l__enumext_miniright_code_viii_box
4726          \end__enumext_mini_page
4727          \par\addvspace{ \g__enumext_minipage_after_skip }
4728        }
4729      \bool_gset_false:N \g__enumext_minipage_active_viii_bool
4730      \bool_gset_true:N \g__enumext_minipage_center_viii_bool
4731      \tl_gclear:N \g__enumext_miniright_code_viii_tl
4732      \dim_gzero:N \g__enumext_minipage_right_viii_dim
4733    }
```

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

13.44 The environment enumext*

enumext*

First we will generate the environment and we will give a temporary definition to __enumext_stop_-item_tmp_vii: equal to __enumext_first_item_tmp_vii: and next to \item equal to __enumext_-start_item_tmp_vii: which we will redefine later. Unlike the implementation used by the shortlst

package, we will not set the values of \rightskip and \@rightskip equal to \@flushglue whose value is 0.0pt plus 1.0 fil, in the tests I have performed this fails in some circumstances and different results are obtained when using pdfTeX and LuaTeX.

```
4734  \NewDocumentEnvironment{enumext*}{ o }
4735    {
4736      \__enumext_safe_exec_vii:
4737      \__enumext_parse_keys_vii:n {#1}
4738      \__enumext_before_list_vii:
4739      \__enumext_start_store_level_vii:
4740      \__enumext_start_list:nn { }
4741        {
4742          \__enumext_list_arg_two_vii:
4743          \__enumext_before_keys_exec_vii:
4744        }
4745      \IfDocumentMetadataTF { \tag_suspend:n {enumext*} } { }
4746      \__enumext_starred_columns_set_vii:
4747      \item[] \scan_stop:
4748      \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_first_item_tmp_vii:
4749      \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
4750      \ignorespaces
4751    }
4752    {
4753      \IfDocumentMetadataTF { \tag_struct_end:n {tag=text-unit} } { }
4754      \__enumext_stop_item_tmp_vii:
4755      \__enumext_remove_extra_parsep_vii:
4756      \__enumext_after_list_vii:
4757    }
```

(*End of definition for* enumext*. *This function is documented on page 5.*)

__enumext_safe_exec_vii: We will first call the function __enumext_is_not_nested: which sets \g__enumext_starred_bool to true if we are NOT nested within enumext, then call the function __enumext_internal_mini_page: to create the environment __enumext_mini_page, we will increment \l__enumext_level_h_int to restrict nesting of the environment, set \l__enumext_starred_bool to true and finally call the function __enumext_is_on_first_level: which sets \l__enumext_starred_first_bool to true if we are not nested, allowing the *"storage system"* to be used.

```
4758  \cs_new_protected:Nn \__enumext_safe_exec_vii:
4759    {
4760      \__enumext_is_not_nested:
4761      \__enumext_internal_mini_page:
4762      \int_incr:N \l__enumext_level_h_int
4763      \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
4764        {
4765          \msg_error:nn { enumext } { nested }
4766        }
4767      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
4768        {
4769          \msg_error:nnn { enumext } { nested-horizontal } { keyans* }
4770        }
4771      \bool_set_true:N \l__enumext_starred_bool
4772      \bool_set_false:N \l__enumext_standar_bool
4773      \__enumext_is_on_first_level:
4774    }
```

(*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.

```
4775  \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
4776    {
4777      \tl_if_novalue:nF {#1}
4778        {
4779          \str_clear:N \l__enumext_series_str
4780          \keys_set:nn { enumext / enumext* } {#1}
4781          \__enumext_parse_series:n {#1}
4782          \__enumext_store_active_keys_vii:n {#1}
4783        }
4784    }
```

(*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*.

```
4785 \cs_new_protected:Nn \__enumext_before_list_vii:
4786   {
4787     \__enumext_vspace_above_vii:
4788     \__enumext_check_ans_active:
4789     \__enumext_before_args_exec_vii:
4790     \__enumext_start_mini_vii:
4791   }
```

(*End of definition for* __enumext_before_list_vii:.)

__enumext_after_list_vii:

The function __enumext_after_list_vii: first calls the function __enumext_stop_mini_vii: which internally calls __enumext_stop_list: and __enumext_stop_store_level_vii: (§13.43.3) used by the keys mini-env, mini-right and mini-right*, then to the functions __enumext_after_stop_-list_vii: used by the key after, __enumext_check_ans_key_hook: used by the key check-ans, __enumext_vspace_below_vii: used by the keys below and below*. Finally set \l__enumext_-starred_bool to false and call the __enumext_resume_save_counter: function used by the series, resume and resume* keys.

```
4792 \cs_new_protected:Nn \__enumext_after_list_vii:
4793   {
4794     \__enumext_stop_mini_vii:
4795     \__enumext_after_stop_list_vii:
4796     \__enumext_check_ans_key_hook:
4797     \__enumext_vspace_below_vii:
4798     \bool_set_false:N \l__enumext_starred_bool
4799     \__enumext_resume_save_counter:
4800   }
```

(*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 *"storing structure"* mechanism in *sequence* for \anskey command and anskey* environment if enumext* are nested in enumext.

```
4801 \cs_new_protected:Nn \__enumext_start_store_level_vii:
4802   {
4803     \bool_if:NT \l__enumext_store_active_bool
4804       {
4805         \int_compare:nNnT { \l__enumext_level_int } > { 0 }
4806           {
4807             \__enumext_store_level_open_vii:
4808           }
4809       }
4810   }
4811 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
4812   {
4813     \bool_if:NT \l__enumext_store_active_bool
4814       {
4815         \int_compare:nNnT { \l__enumext_level_int } > { 0 }
4816           {
4817             \__enumext_store_level_close_vii:
4818           }
4819       }
4820   }
```

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

13.44.1 The command \item in enumext*

__enumext_first_item_tmp_vii:

The __enumext_first_item_tmp_vii: function will remove horizontal space equal to \labelwidth plus \labelsep to the left of the *"first"* \item in the environment at the point of execution of this function, where it is equal to the __enumext_stop_item_tmp_vii: function inside the environment body definition.

```
4821 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_vii:
4822   {
4823     \skip_horizontal:n
4824       {
4825         -\l__enumext_labelwidth_vii_dim - \l__enumext_labelsep_vii_dim
```

```
4826              }
4827          \ignorespaces
4828      }
```

(*End of definition for* __enumext_first_item_tmp_vii:.)

__enumext_start_item_tmp_vii:
__enumext_item_peek_args_vii:
__enumext_joined_item_vii:w
__enumext_standar_item_vii:w
__enumext_starred_item_vii:w
__enumext_starred_item_vii_aux_i:w
__enumext_starred_item_vii_aux_ii:w
__enumext_starred_item_vii_aux_iii:w

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

```
4829 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
4830    {
4831        \__enumext_stop_item_tmp_vii:
4832        \int_incr:N \l__enumext_item_column_pos_vii_int
4833        \int_gincr:N \g__enumext_item_count_all_vii_int
4834        \__enumext_item_peek_args_vii:
4835    }
```

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).

```
4836 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
4837    {
4838        \peek_meaning:NTF (
4839            { \__enumext_joined_item_vii:w }
4840            { \__enumext_joined_item_vii:w (1) }
4841    }
```

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.

```
4842 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
4843    {
4844        \__enumext_starred_joined_item_vii:n {#1}
4845        \peek_meaning_remove:NTF *
4846            { \__enumext_starred_item_vii:w  }
4847            { \__enumext_standar_item_vii:w }
4848    }
```

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].

```
4849 \cs_new_protected:Npn \__enumext_standar_item_vii:w
4850    {
4851        \bool_set_false:N \l__enumext_item_starred_vii_bool
4852        \peek_meaning:NTF [
4853            {
4854                \bool_set_eq:NN \l__enumext_wrap_label_vii_bool \l__enumext_wrap_label_opt_vii_bool
4855                \__enumext_start_item_vii:w
4856            }
4857            {
4858                \bool_set_true:N \l__enumext_wrap_label_vii_bool
4859                \legacy_if_set_true:n { @noitemarg }
4860                \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ] \ignorespaces
4861            }
4862    }
```

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*⟩].

```
4863 \cs_new_protected:Npn \__enumext_starred_item_vii:w
4864    {
4865        \bool_set_true:N \l__enumext_item_starred_vii_bool
4866        \bool_set_true:N \l__enumext_wrap_label_vii_bool
4867        \peek_meaning:NTF [
4868            { \__enumext_starred_item_vii_aux_i:w }
```

```
4869            { \__enumext_starred_item_vii_aux_ii:w }
4870        }
4871    \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
4872        {
4873            \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
4874            \__enumext_starred_item_vii_aux_ii:w
4875        }
4876    \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
4877        {
4878            \peek_meaning:NTF [
4879                { \__enumext_starred_item_vii_aux_iii:w }
4880                {
4881                    \dim_set_eq:NN \l__enumext_item_symbol_sep_vii_dim \l__enumext_labelsep_vii_dim
4882                    \legacy_if_set_true:n { @noitemarg }
4883                    \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ] \ignorespaces
4884                }
4885        }
4886    \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
4887        {
4888            \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
4889            \legacy_if_set_true:n { @noitemarg }
4890            \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ] \ignorespaces
4891        }
```

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

__enumext_fake_make_label_vii:n   The __enumext_fake_make_label_vii:n function will be in charge of handling our definition of \item. First we increment the counter enumXvii for the enumerated items and activate support for the *check answers* mechanism, followed by support for \item*[⟨*symbol*⟩][⟨*offset*⟩] if present, then the wrap-label and wrap-label* keys which we execute using \makebox whose width will be given by the labelwidth key and position by the align key, inside the argument of this we will execute the font key together with the function defined by the wrap-label or wrap-label* keys. Finally we execute the labelsep key applying a \skip_horizontal:N and \ignorespaces.

🔷 For compatibility with *tagged* PDF and hyperref when an environment enumext is nested in enumext* and the key save-ans is not active need setting the \if@hyper@item switch to *"true"*. The explanation for this is given by the master Heiko Oberdiek on \refstepcounter{enumi} twice (or more) creates destination with the same identifier. This patch is only needed if you are running pdflatex and not if you are running lualatex

```
4892    \cs_new_protected_nopar:Npn \__enumext_fake_make_label_vii:n #1
4893        {
4894            \legacy_if:nT { @noitemarg }
4895                {
4896                    \legacy_if_set_false:n { @noitemarg }
4897                    \legacy_if:nT { @nmbrlist }
4898                        {
4899                            \IfDocumentMetadataTF
4900                                {
4901                                    \bool_if:NT \l__enumext_hyperref_bool
4902                                        {
4903                                            \legacy_if_set_true:n { @hyper@item }
4904                                        }
4905                                } { }
4906                            \refstepcounter{enumXvii}
4907                            \bool_if:NT \l__enumext_check_answers_bool
4908                                {
4909                                    \int_gincr:N \g__enumext_item_number_int
4910                                    \bool_set_true:N \l__enumext_item_number_bool
4911                                }
4912                        }
4913                }
4914            \bool_if:NT \l__enumext_item_starred_vii_bool
4915                {
4916                    \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
4917                        {
4918                            \tl_gset_eq:NN
4919                                \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
4920                        }
4921                    \mode_leave_vertical:
4922                    \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
4923                    \hbox_overlap_left:n { \g__enumext_item_symbol_aux_vii_tl }
4924                    \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
```

```
4925          \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
4926        }
4927      \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
4928        {
4929          \tl_use:N \l__enumext_label_font_style_vii_tl
4930          \bool_if:NTF \l__enumext_wrap_label_vii_bool
4931            {
4932              \__enumext_wrapper_label_vii:n {#1}
4933            }
4934            { #1 }
4935        }
4936      \skip_horizontal:N \l__enumext_labelsep_vii_dim \ignorespaces
4937    }
```

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

13.44.2 **Real definition of** `\item` **in enumext***

The functions `\__enumext_start_item_vii:w` and `\__enumext_stop_item_vii:` executing the true definition of `\item` inside the `enumext*` environment, unlike the implementation in `shortlst` we will NOT use an extra group and the plain form of the `lrbox` environment.

`\__enumext_start_item_vii:w`
`\__enumext_stop_item_vii:`

The first thing we will do is set the value of `\__enumext_stop_item_tmp_vii:` equal to `\__enumext_-stop_item_vii:` which we will define later, after that we will start capturing `\item` and *"item content"* in a *horizontal box* where the width will be `\itemwidth` plus `\labelwidth` plus `\labelsep`.

```
4938  \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
4939    {
4940      \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
4941      \hbox_set_to_wd:Nnw \l__enumext_item_text_vii_box
4942        {
4943          \l__enumext_joined_width_vii_dim
4944          + \l__enumext_labelwidth_vii_dim
4945          + \l__enumext_labelsep_vii_dim
4946        }
```

Redefine the `\footnote` command.

```
4947        \__enumext_renew_footnote_starred:
```

Now we insert our *sockets* for *tagging* PDF support and run `\item`.

```
4948        \__enumext_start_list_tag:n {enumext*}
4949        \__enumext_fake_make_label_vii:n {#1}
4950        \__enumext_stop_start_list_tag:
```

Finally we open the `minipage` environment, capture the *"item content"*, make `\parindent` take the value of the key `listparindent` and `\parskip` take the value of the key `parsep`, then execute the keys `itemindent` and `first`.

🎯 Here the use of `\unskip` and `\skip_horizontal:n` with the value of `listparindent` is necessary, otherwise an unwanted space is created when using `\item[⟨opt⟩]` and the value passed to the key `itemindent` is incremented.

```
4951        \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
4952        \dim_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
4953        \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
4954        \__enumext_unskip_unkern:
4955        \__enumext_unskip_unkern:
4956        \skip_horizontal:n { -\l__enumext_listparindent_vii_dim } \ignorespaces
4957        \tl_use:N \l__enumext_fake_item_indent_vii_tl
4958        \tl_use:N \l__enumext_after_list_args_vii_tl
4959    }
```

The `\__enumext_stop_item_vii:` function will finish the fetching `\item` and *"item content"* by closing the `minipage` environment, the *sockets* for *tagging* PDF and the *horizontal box*.

```
4960  \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
4961    {
4962        \__enumext_endminipage:
4963      \__enumext_stop_list_tag:n {enumext*}
4964      \hbox_set_end:
```

Here we will reduce the *warnings* a bit by setting the value of `\hbadness` to `10000`, print `\item` and *"item content"* from the *horizontal box*,.

```
4965      \int_set:Nn \hbadness { 10000 }
4966      \box_use_drop:N \l__enumext_item_text_vii_box
```

Finally apply the *vertical space* between rows set by `itemsep` key passed to `\parsep` using `\par\noindent` and *horizontal space* between columns set by `columns-sep` key using `\skip_horizontal:N`.

```
4967    \int_compare:nNnTF
4968      { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
4969      {
4970        \par\noindent
4971        \int_zero:N \l__enumext_item_column_pos_vii_int
4972      }
4973      {
4974        \skip_horizontal:N \l__enumext_columns_sep_vii_dim
4975      }
4976    }
```

(*End of definition for* `\__enumext_start_item_vii:w` *and* `\__enumext_stop_item_vii:`.)

`\__enumext_remove_extra_parsep_vii:`

Remove the extra *vertical space* equal to `\parsep=\itemsep` when the total number of `\item` is divisible by the number of `\item` in the last row of the environment. Here the use of `\unskip` or `\removelastskip` fails and does not obtain the expected result, using `\vspace` is the option and in this case, we can use a simplified version since we are always in ⟨*vertical mode*⟩.

```
4977    \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
4978    {
4979      \int_compare:nNnT
4980        {
4981          \int_mod:nn
4982            {  \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
4983        }
4984        =
4985        { 0 }
4986        {
4987          \para_end:
4988          \skip_vertical:n { -\l__enumext_itemsep_vii_skip }
4989          \skip_vertical:N \c_zero_skip
4990          \int_gzero:N \g__enumext_item_count_all_vii_int
4991        }
4992    }
```

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

As we don't want our check to be executed `check-ans` by levels but on the complete list, we will take it out of the `enumext*` environment using the *"hook"* function `\__enumext_after_env:nn`.

```
4993    \__enumext_after_env:nn {enumext*}
4994    {
4995      \__enumext_execute_after_env:
4996    }
```

13.45   The environment keyans*

keyans*

The implementation of `keyans*` environment is the similar as that used by the `enumext*` environment except for the `\__enumext_check_starred_cmd:n` function added in the second part.

```
4997    \NewDocumentEnvironment{keyans*}{ o }
4998    {
4999      \__enumext_safe_exec_viii:
5000      \__enumext_parse_keys_viii:n {#1}
5001      \__enumext_before_list_viii:
5002      \__enumext_start_list:nn { }
5003        {
5004          \__enumext_list_arg_two_viii:
5005          \__enumext_before_keys_exec_viii:
5006        }
5007      \IfDocumentMetadataTF { \tag_suspend:n {keyans*} } { }
5008      \__enumext_starred_columns_set_viii:
5009      \item[] \scan_stop:
5010      \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_first_item_tmp_viii:
5011      \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
5012      \ignorespaces
5013    }
5014    {
5015      \IfDocumentMetadataTF { \tag_struct_end:n {tag=text-unit} } { }
5016      \__enumext_stop_item_tmp_viii:
5017      \__enumext_remove_extra_parsep_viii:
5018      \__enumext_check_starred_cmd:n { item }
```

```
5019        \__enumext_after_list_viii:
5020      }
```

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

__enumext_safe_exec_viii: The __enumext_safe_exec_viii: function will first check if the save-ans key is active and only when this is true the environment will be available, it will increment the value of \l__enumext_keyans_level_h_int and return an error message when we are nesting the environment, then it will call the __enumext_-keyans_name_and_start: function in charge of saving the name of the environment and the line it is running on, then it will check if we are trying to nest keyans* in enumext* returning an error and we will set \l__enumext_starred_bool to true, finally we will check if we are within the appropriate level within the enumext environment.

```
5021 \cs_new_protected:Nn \__enumext_safe_exec_viii:
5022    {
5023      \bool_if:NF \l__enumext_store_active_bool
5024        {
5025          \msg_error:nnnn { enumext } { wrong-place }{ keyans* }{ save-ans }
5026        }
5027      \int_incr:N \l__enumext_keyans_level_h_int
5028      \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
5029        {
5030          \msg_error:nn { enumext } { nested }
5031        }
5032      \__enumext_keyans_name_and_start:
5033      \bool_if:NT \l__enumext_starred_bool
5034        {
5035          \msg_error:nnn { enumext } { nested-horizontal } { enumext* }
5036        }
5037      \bool_set_true:N \l__enumext_starred_bool
5038      % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
5039      \bool_set_false:N \l__enumext_store_active_bool
5040      \int_compare:nNnT { \l__enumext_level_int } > { 1 }
5041        {
5042          \msg_error:nn { enumext } { keyans-wrong-level }
5043        }
5044    }
```

(*End of definition for* __enumext_safe_exec_viii:.)

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

```
5045 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
5046    {
5047      \tl_if_novalue:nF {#1}
5048        {
5049          \keys_set:nn { enumext / keyans* } {#1}
5050        }
5051    }
```

(*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.

```
5052 \cs_new_protected:Nn \__enumext_before_list_viii:
5053    {
5054      \__enumext_vspace_above_viii:
5055      \__enumext_before_args_exec_viii:
5056      \__enumext_start_mini_viii:
5057    }
```

(*End of definition for* __enumext_before_list_viii:.)

__enumext_after_list_viii: The function __enumext_after_list_viii: first call the function __enumext_stop_mini_viii:, then apply the {⟨*code*⟩} handled by the after key together with the *vertical space* handled by the below key if they are present.

```
5058 \cs_new_protected:Nn \__enumext_after_list_viii:
5059    {
5060      \__enumext_stop_mini_viii:
5061      \__enumext_after_stop_list_viii:
5062      \__enumext_vspace_below_viii:
5063    }
```

(*End of definition for* __enumext_after_list_viii:.)

13.45.1 The command `\item` in keyans*

The idea here is to make the `\item` command behave in the same way as in the keyans environment with the difference of the *optional argument* (⟨*number*⟩) which works in the same way as in the enumext* environment. In simple terms we want to store the ⟨*label*⟩ next to the [⟨*content*⟩] if it is present in the *sequence* and *prop list* defined by save-ans key for `\item*`, `\item*[`⟨*content*⟩`]`, `\item(`⟨*number*⟩`)*` and `\item(`⟨*number*⟩`)*[`⟨*content*⟩`]` commands.

`\__enumext_first_item_tmp_viii:` The `\__enumext_first_item_tmp_viii:` function will remove horizontal space equal to `\labelwidth` plus `\labelsep` to the left of the *"first"* `\item` in the environment at the point of execution of this function, where it is equal to the `\__enumext_stop_item_tmp_viii:` function inside the environment body definition.

```
5064 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_viii:
5065   {
5066     \skip_horizontal:n
5067       {
5068         -\l__enumext_labelwidth_viii_dim - \l__enumext_labelsep_viii_dim
5069       }
5070     \ignorespaces
5071   }
```

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

`\__enumext_start_item_tmp_viii:`
`\__enumext_item_peek_args_viii:`
`\__enumext_joined_item_viii:w`
`\__enumext_standar_item_viii:w`

First we will call the function `\__enumext_stop_item_tmp_viii:` that we will redefine later, we will increment the value of `\l__enumext_item_column_pos_viii_int` that will count the item's by rows and the value of `\g__enumext_item_count_all_viii_int` that will count the total of item's in the environment. After that we will call the function `\__enumext_item_peek_args_viii:` that will handle the arguments passed to `\item`.

```
5072 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
5073   {
5074     \__enumext_stop_item_tmp_viii:
5075     \int_incr:N \l__enumext_item_column_pos_viii_int
5076     \int_gincr:N \g__enumext_item_count_all_viii_int
5077     \__enumext_item_peek_args_viii:
5078   }
```

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`).

```
5079 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
5080   {
5081     \peek_meaning:NTF (
5082       { \__enumext_joined_item_viii:w }
5083       { \__enumext_joined_item_viii:w (1) }
5084   }
```

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`.

```
5085 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
5086   {
5087     \__enumext_starred_joined_item_viii:n {#1}
5088     \peek_meaning_remove:NTF *
5089       { \__enumext_starred_item_viii:w }
5090       { \__enumext_standar_item_viii:w }
5091   }
```

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 ]`.

```
5092 \cs_new_protected:Npn \__enumext_standar_item_viii:w
5093   {
5094     \bool_set_false:N \l__enumext_item_starred_viii_bool
5095     \bool_set_false:N \l__enumext_item_wrap_key_bool
5096     \peek_meaning:NTF [
5097       {
5098         \bool_set_eq:NN \l__enumext_wrap_label_viii_bool \l__enumext_wrap_label_opt_viii_bool
```

```
5099          \__enumext_start_item_viii:w
5100        }
5101      {
5102        \bool_set_true:N \l__enumext_wrap_label_viii_bool
5103        \legacy_if_set_true:n { @noitemarg }
5104        \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ] \ignorespaces
5105      }
5106    }
```

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

__enumext_starred_item_viii:w

__enumext_starred_item_viii_aux_i:w

__enumext_starred_item_viii_aux_ii:w

__enumext_keyans_starred_item_star:

The function __enumext_starred_item_viii:w together with the specified auxiliary functions aux_i:w and aux_ii:w execute \item* and \item*[⟨content⟩].

```
5107  \cs_new_protected:Npn \__enumext_starred_item_viii:w
5108    {
5109      \bool_set_true:N \l__enumext_item_starred_viii_bool
5110      \bool_set_true:N \l__enumext_item_wrap_key_bool
5111      \bool_set_true:N \l__enumext_wrap_label_viii_bool
5112      \peek_meaning:NTF [
5113        { \__enumext_starred_item_viii_aux_i:w }
5114        { \__enumext_starred_item_viii_aux_ii:w }
5115    }
```

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.

```
5116  \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
5117    {
5118      \tl_clear:N \l__enumext_store_current_label_tl
5119      \tl_if_novalue:nF { #1 }
5120        {
5121          \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_viii_tl
5122            {
5123              \tl_put_right:NV \l__enumext_store_current_label_tl \l__enumext_store_keyans_item_opt_
5124              \tl_put_right:Nn \l__enumext_store_current_label_tl { #1 }
5125            }
5126          \tl_set:Nn \l__enumext_store_current_opt_arg_tl { #1 }
5127        }
5128      \__enumext_starred_item_viii_aux_ii:w
5129    }
5130  \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
5131    {
5132      \legacy_if_set_true:n { @noitemarg }
5133      \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ] \ignorespaces
5134    }
```

The function __enumext_keyans_starred_item_star: will be in charge of storing the current ⟨label⟩ for \item* followed by the [⟨content⟩] for \item*[⟨content⟩] if present in the *sequence* and *prop list* set by the save-ans key. In this same function the keys show-ans, show-pos, mark-sep and save-ref are implemented.

```
5135  \cs_new_protected:Nn \__enumext_keyans_starred_item_star:
5136    {
5137      \tl_put_left:Ne \l__enumext_store_current_label_tl { \l__enumext_label_viii_tl }
5138      \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
5139      \__enumext_keyans_store_ref:
5140      \tl_put_left:Nn \l__enumext_store_current_label_tl { \item }
5141      \__enumext_keyans_addto_seq_link:
5142      \int_gincr:N \g__enumext_check_starred_cmd_int
5143      \dim_compare:nNnT { \l__enumext_mark_sym_sep_viii_dim } = { \c_zero_dim }
5144        {
5145          \dim_set:Nn \l__enumext_mark_sym_sep_viii_dim { \l__enumext_labelsep_viii_dim }
5146        }
5147      \bool_if:NT \l__enumext_show_answer_bool
5148        {
5149          \tl_set_eq:NN \l__enumext_mark_answer_sym_tl \l__enumext_mark_answer_sym_viii_tl
5150          \str_set_eq:NN \l__enumext_mark_position_str \l__enumext_mark_position_viii_str
5151          \__enumext_print_keyans_box:NN
5152            \l__enumext_labelwidth_viii_dim \l__enumext_mark_sym_sep_viii_dim
5153        }
5154      \bool_if:NT \l__enumext_show_position_bool
```

```
5155        {
5156            \tl_set:Ne \l__enumext_mark_answer_sym_tl
5157              {
5158                \group_begin:
5159                  \exp_not:N \normalfont
5160                  \exp_not:N \footnotesize [ \int_eval:n
5161                    {
5162                        \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
5163                    }
5164                    ]
5165                \group_end:
5166              }
5167            \str_set_eq:NN \l__enumext_mark_position_str \l__enumext_mark_position_viii_str
5168            \__enumext_print_keyans_box:NN
5169                \l__enumext_labelwidth_viii_dim \l__enumext_mark_sym_sep_viii_dim
5170        }
5171    }
```

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

`\__enumext_keyans_wraper_label_viii:n`
`\__enumext_fake_make_label_viii:n`

The implementation at this is very similar to that of the enumext* environment.

```
5172 \cs_new_protected:Npn \__enumext_keyans_wraper_label_viii:n #1
5173    {
5174        \bool_lazy_all:nT
5175          {
5176            { \bool_if_p:N \l__enumext_wrap_label_viii_bool          }
5177            { \bool_if_p:N \l__enumext_show_answer_bool              }
5178            { \bool_if_p:N \l__enumext_item_wrap_key_bool           }
5179            { \cs_if_exist_p:N \__enumext_keyans_wrapper_item_viii:n }
5180          }
5181          {
5182            \cs_set_eq:NN
5183                \__enumext_wrapper_label_viii:n \__enumext_keyans_wrapper_item_viii:n
5184          }
5185        \bool_if:NTF \l__enumext_wrap_label_viii_bool
5186          {
5187            \__enumext_wrapper_label_viii:n {#1}
5188          }
5189          { #1 }
5190    }
5191 \cs_new_protected_nopar:Npn \__enumext_fake_make_label_viii:n #1
5192    {
5193        \legacy_if:nT { @noitemarg }
5194          {
5195            \legacy_if_set_false:n { @noitemarg }
5196            \legacy_if:nT { @nmbrlist }
5197              {
5198                \refstepcounter{enumXviii}
5199              }
5200          }
5201        \bool_if:NT \l__enumext_item_starred_viii_bool
5202          {
5203            \__enumext_keyans_starred_item_star:
5204          }
5205        \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
5206          {
5207            \tl_use:N \l__enumext_label_font_style_viii_tl
5208            \__enumext_keyans_wraper_label_viii:n {#1}
5209          }
5210        \skip_horizontal:N \l__enumext_labelsep_viii_dim \ignorespaces
5211    }
```

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

13.45.2 Real definition of \item in keyans*

The implementation at this is very similar to that of the enumext* environment.

`\__enumext_start_item_viii:w`
`\__enumext_stop_item_viii:`

```
5212 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [ #1]
5213    {
5214        \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
5215        \hbox_set_to_wd:Nnw \l__enumext_item_text_viii_box
5216          {
```

```
5217            \l__enumext_joined_width_viii_dim
5218          + \l__enumext_labelwidth_viii_dim
5219          + \l__enumext_labelsep_viii_dim
5220          }
5221        \__enumext_renew_footnote_starred:
5222        \__enumext_start_list_tag:n {keyans*}
5223        \__enumext_fake_make_label_viii:n {#1}
5224        \__enumext_stop_start_list_tag:
5225        \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim  }
5226          \dim_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
5227          \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
5228          \__enumext_unskip_unkern:
5229          \__enumext_unskip_unkern:
5230          \skip_horizontal:n { -\l__enumext_listparindent_viii_dim } \ignorespaces
5231          \tl_use:N \l__enumext_fake_item_indent_viii_tl
5232          \bool_if:NT \l__enumext_item_starred_viii_bool
5233            {
5234              \__enumext_keyans_show_item_opt_viii:
5235            }
5236          \tl_use:N \l__enumext_after_list_args_viii_tl
5237      }
5238  \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
5239    {
5240        \__enumext_endminipage:
5241      \__enumext_stop_list_tag:n {keyans*}
5242      \hbox_set_end:
5243      \int_set:Nn \hbadness { 10000 }
5244      \box_use_drop:N \l__enumext_item_text_viii_box
5245      \int_compare:nNnTF
5246        { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
5247        {
5248          \par\noindent
5249          \int_zero:N \l__enumext_item_column_pos_viii_int
5250        }
5251        {
5252          \skip_horizontal:N \l__enumext_columns_sep_viii_dim
5253        }
5254    }
```

(*End of definition for* __enumext_start_item_viii:w *and* __enumext_stop_item_viii:.)

__enumext_remove_extra_parsep_viii: The implementation at this is very similar to that of the enumext* environment.

```
5255  \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
5256    {
5257      \int_compare:nNnT
5258        {
5259          \int_mod:nn
5260            { \g__enumext_item_count_all_viii_int }
5261            { \l__enumext_columns_viii_int }
5262        }
5263        =
5264        { 0 }
5265        {
5266          \para_end:
5267          \skip_vertical:n { -\l__enumext_itemsep_viii_skip }
5268          \skip_vertical:N \c_zero_skip
5269          \int_gzero:N \g__enumext_item_count_all_viii_int
5270        }
5271    }
```

(*End of definition for* __enumext_remove_extra_parsep_viii:.)

13.46 The command \getkeyans

\getkeyans
__enumext_getkeyans_aux:n
__enumext_getkeyans:nn

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

```
5272  \NewDocumentCommand \getkeyans { m }
5273    {
5274      \exp_args:Ne \__enumext_getkeyans_aux:n
5275        { \tl_to_str:e { \text_expand:n {#1} } }
5276    }
```

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

```
5277 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
5278   {
5279     \str_if_in:nnTF {#1} { : }
5280       {
5281         \use:e
5282           {
5283             \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
5284               { {##1} {##2} }
5285           }
5286         \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
5287       }
5288     { \msg_error:nnn { enumext } { missing-colon } {#1} }
5289   }
```

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

```
5290 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
5291   {
5292     \prop_if_exist:cTF { g__enumext_#1_prop }
5293       {
5294         \prop_item:cn { g__enumext_#1_prop }{#2}
5295       }
5296       {
5297         \msg_error:nnn { enumext } { undefined-storage-anskey } {#1}
5298       }
5299   }
```

(*End of definition for* \getkeyans, __enumext_getkeyans_aux:n, *and* __enumext_getkeyans:nn. *This function is documented on page 18.*)

13.47   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*⟩].

```
5300 \keys_define:nn  { enumext / print }
5301   {
5302     print*  .code:n    = \keys_precompile:neN { enumext / enumext* }
5303                           { \__enumext_filter_save_key:n {#1} }
5304                           \l__enumext_print_keyans_starred_tl, % starred cmd
5305     print*  .initial:n = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5306                             rightmargin=0pt, listparindent=0pt, nosep, label=\arabic*.,
5307                             columns=2, first=\small, font=\small },
5308     print-1 .code:n    = \keys_precompile:neN { enumext / level-1 }
5309                           { \__enumext_filter_save_key:n {#1} }
5310                           \l__enumext_print_keyans_i_tl,
5311     print-1 .initial:n = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5312                             rightmargin=0pt, listparindent=0pt, nosep, label=\arabic*.,
5313                             columns=2, first=\small, font=\small },
5314     print-2 .code:n    = \keys_precompile:neN { enumext / level-2 }
5315                           { \__enumext_filter_save_key:n {#1} }
5316                           \l__enumext_print_keyans_ii_tl,
5317     print-2 .initial:n = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5318                             rightmargin=0pt, listparindent=0pt,  nosep, label=(\alph*),
5319                             first=\small, font=\small },
5320     print-3 .code:n    = \keys_precompile:neN { enumext / level-3 }
5321                           { \__enumext_filter_save_key:n {#1} }
5322                           \l__enumext_print_keyans_iii_tl,
5323     print-3 .initial:n = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5324                             rightmargin=0pt, listparindent=0pt, nosep, label=\roman*.,
5325                             first=\small, font=\small },
5326     print-4 .code:n    = \keys_precompile:neN { enumext / level-4 }
```

```
5327                                         { \__enumext_filter_save_key:n {#1} }
5328                                         \l__enumext_print_keyans_iv_tl,
5329       print-4 .initial:n  = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5330                                         rightmargin=0pt, listparindent=0pt, nosep, label=\Alph*.,
5331                                         first=\small, font=\small },
5332       print-* .code:n      = \keys_precompile:neN { enumext / enumext* }
5333                                         { \__enumext_filter_save_key:n {#1} }
5334                                         \l__enumext_print_keyans_vii_tl, % starred nested
5335       print-* .initial:n  = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5336                                         rightmargin=0pt, listparindent=0pt, nosep, label=\arabic*.,
5337                                         first=\small, font=\small },
5338       }
```

🖋 The reason for storing ⟨keys⟩ in token lists using \keys_precompile:neN is because the keys are set via \setenumext but are later executed by running the command \printkeyans and they are not handled directly by its *optional argument*, except those related to the *first* opening level.

\printkeyans
__enumext_printkeyans:nnn

Create a user command to print *"all stored content"* in *sequence* for \anskey, anskey*, \item* and \anspic*. Within a group we will run our *"precompiled keys"* and then call the internal function __enumext_-printkeyans:nnn.

```
5339   \NewDocumentCommand \printkeyans { s O{} m }
5340     {
5341       \group_begin:
5342         \tl_use:N \l__enumext_print_keyans_i_tl
5343         \tl_use:N \l__enumext_print_keyans_ii_tl
5344         \tl_use:N \l__enumext_print_keyans_iii_tl
5345         \tl_use:N \l__enumext_print_keyans_iv_tl
5346         \tl_use:N \l__enumext_print_keyans_vii_tl
5347         \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
5348       \group_end:
5349     }
```

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.

```
5350   \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
5351     {
5352       \seq_if_exist:cTF { g__enumext_#3_seq }
5353         {
5354           \seq_if_empty:cF { g__enumext_#3_seq }
5355             {
```

If the *starred argument* '*' is present we will check that the environment enumext* is not saved in the *sequence*, then execute the variable \l__enumext_print_keyans_starred_tl that contains the default ⟨keys⟩ for the environment enumext*, we set \l__enumext_base_line_fix_bool and \l__enumext_print_keyans_-star_bool to true for *baseline correction*, open the enumext* environment passing the *optional argument* and map the *sequence*, then set \l__enumext_base_line_fix_bool and \l__enumext_print_keyans_-star_bool to false.

```
5356                 \bool_if:nTF {#1}
5357                   {
5358                     \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
5359                       {
5360                         \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
5361                       }
5362                       {
5363                         \tl_use:N \l__enumext_print_keyans_starred_tl
5364                         \bool_set_true:N \l__enumext_base_line_fix_bool
5365                         \bool_set_true:N \l__enumext_print_keyans_star_bool
5366                         \begin{enumext*}[#2]
5367                           \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5368                         \end{enumext*}
5369                         \bool_set_false:N \l__enumext_base_line_fix_bool
5370                         \bool_set_false:N \l__enumext_print_keyans_star_bool
5371                       }
5372                   }
```

Otherwise it will open the environment enumext passing the *optional argument* to the *"first level"* then map the *sequence*.

```
5373                   {
5374                     \begin{enumext}[#2]
5375                       \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5376                     \end{enumext}
5377                   }
```

```
5378                }
5379            }
5380            {
5381                \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
5382            }
5383        }
```

(*End of definition for* \printkeyans *and* __enumext_printkeyans:nnn. *This function is documented on page* *19*.)

13.48   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.

```
5384  \cs_new:Npn \__enumext_filter_first_level:n #1
5385    {
5386      \use:e
5387        {
5388          \keyval_parse:NNn
5389            \__enumext_filter_first_level_key:n
5390            \__enumext_filter_first_level_pair:nn {#1}
5391        }
5392    }
```

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*.

```
5393  \cs_new:Npn \__enumext_filter_first_level_key:n #1
5394    {
5395      \str_case:nnF {#1}
5396        {
5397          { resume   } {}
5398          { resume*  } {}
5399        }
5400        { , { \exp_not:n {#1} } }
5401    }
```

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.

```
5402  \cs_new:Npn \__enumext_filter_first_level_pair:nn #1#2
5403    {
5404      \str_case:nnF {#1}
5405        {
5406          { series } {}
5407          { resume } {}
5408          { save-ans } {}
5409        }
5410        { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
5411    }
```

(*End of definition for* __enumext_filter_first_level:n, __enumext_filter_first_level_key:n, *and* __enumext_fil-
ter_first_level_pair:nn.)

Now define a *"meta families"* of ⟨keys⟩ to access from \setenumext.

```
5412  \keys_define:nn { enumext / meta-families }
5413    {
5414      enumext-1 .code:n =
5415              {
5416                \keys_set:ne { enumext / level-1 }
5417                  {
5418                    \__enumext_filter_first_level:n {#1}
5419                  }
5420              } ,
5421      enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
5422      enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
5423      enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
5424      keyans    .code:n = { \keys_set:nn { enumext / keyans  } {#1} } ,
5425      enumext*  .code:n =
5426              {
5427                \keys_set:ne { enumext / enumext* }
5428                  {
```

```
5429                         \__enumext_filter_first_level:n {#1}
5430                       }
5431               } ,
5432     keyans*   .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
5433     print*    .code:n = { \keys_set:nn { enumext / print   } { print*  = {#1} } } ,
5434     print-1   .code:n = { \keys_set:nn { enumext / print   } { print-1 = {#1} } } ,
5435     print-2   .code:n = { \keys_set:nn { enumext / print   } { print-2 = {#1} } } ,
5436     print-3   .code:n = { \keys_set:nn { enumext / print   } { print-3 = {#1} } } ,
5437     print-4   .code:n = { \keys_set:nn { enumext / print   } { print-4 = {#1} } } ,
5438     print-*   .code:n = { \keys_set:nn { enumext / print   } { print-* = {#1} } } ,
5439     unknown   .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
5440   }
```

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

```
5441 \seq_const_from_clist:Nn \c__enumext_all_families_seq
5442   {
5443     enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
5444     keyans*, print-1, print-2, print-3, print-4, print-*, print*,
5445   }
```

\setenumext   Now we define the user command \setenumext.
__enumext_set_parse:n
__enumext_set_error:nn

```
5446 \NewDocumentCommand \setenumext { O{enumext,1} +m }
5447   {
5448     \seq_clear:N \l__enumext_setkey_tmpa_seq
5449     \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
5450     \int_set:Nn \l__enumext_setkey_tmpa_int
5451       {
5452         \seq_count:N \l__enumext_setkey_tmpb_seq
5453       }
5454     \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
5455       {
5456         \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
5457         \seq_map_function:NN \l__enumext_setkey_tmpb_seq \__enumext_set_parse:n
5458         \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
5459           {
5460             \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
5461           }
5462       }
5463       {
5464         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
5465       }
5466     \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
5467       { \seq_map_inline:Nn \c__enumext_all_families_seq }
5468       { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
5469       {
5470         \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
5471       }
5472   }
```

Internal functions used by the \setenumext command.

```
5473 \cs_new_protected:Npn \__enumext_set_parse:n #1
5474   {
5475     \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
5476     \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
5477       { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
5478     \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
5479       {
5480         \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
5481           { \tl_trim_spaces:n {#1} }
5482       }
5483       { \__enumext_set_error:nn {#1} { } }
5484   }
5485 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
5486   { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }
```

(*End of definition for* \setenumext *,* __enumext_set_parse:n *, and* __enumext_set_error:nn*. This function is documented on page 6.*)

13.49   The command \setenumextmeta

The command \setenumextmeta will be responsible for adding new *"meta-keys"* for the enumext and enumext* environments. The implementation code was given by Jonathan P. Spratte (@Skillmon) answer in Add .meta key to existing keys (l3keys).

First we will create a prop list `\c__enumext_meta_paths_prop` to handle the *optional argument*.

```
5487 \prop_const_from_keyval:Nn \c__enumext_meta_paths_prop
5488   {
5489     {enumext,1} = level-1,
5490     {enumext,2} = level-2,
5491     {enumext,3} = level-3,
5492     {enumext,4} = level-4,
5493     {enumext*}  = enumext*
5494   }
```

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

```
5495 \NewDocumentCommand \setenumextmeta { s O{enumext,1} m +m }
5496   {
5497     \str_if_eq:eeTF { \tl_trim_spaces:n {#3} } { unknown }
5498       { \msg_error:nn { enumext } { prohibited-unknown } }
5499       {
5500         \bool_if:nTF {#1}
5501           {
5502             \int_step_inline:nn { 4 }
5503               { \__enumext_add_meta_key:nnn { enumext, ##1 } {#3} {#4} }
5504             \__enumext_add_meta_key:nnn { enumext* } {#3} {#4}
5505           }
5506           { \__enumext_add_meta_key:nnn {#2} {#3} {#4} }
5507       }
5508   }
```

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

```
5509 \cs_new_protected:Npn \__enumext_add_meta_key:nnn #1
5510   {
5511     \tl_set:Nn \l__enumext_meta_path_tl {#1}
5512     \tl_replace_all:Nnn \l__enumext_meta_path_tl { ~ } {}
5513     \prop_get:NVNTF
5514       \c__enumext_meta_paths_prop \l__enumext_meta_path_tl \l__enumext_meta_path_tl
5515       { \__enumext_def_meta_key:Vnn \l__enumext_meta_path_tl }
5516       {
5517         \msg_error:nnn { enumext } { unknown-set } {#1}
5518         \use_none:nn
5519       }
5520   }
5521 \cs_new_protected:Npn \__enumext_def_meta_key:nnn #1#2#3
5522   {
5523     \bool_lazy_or:nnTF
5524       { \keys_if_exist_p:nn { enumext / #1 } {#2} }
5525       { \keys_if_exist_p:nn { enumext / enumext* } {#2} }
5526       { \msg_error:nnn { enumext } { already-defined } {#2} }
5527       {
5528         \keys_define:nn { enumext / #1 }
5529           {
5530             #2 .meta:n = {#3},
5531             #2 .value_forbidden:n = true
5532           }
5533       }
5534   }
5535 \cs_generate_variant:Nn \__enumext_def_meta_key:nnn { V }
```

(*End of definition for* `\setenumextmeta` *and others. This function is documented on page 6.*)

13.50   The command `\foreachkeyans`

The command `\foreachkeyans` will execute a *loop* over the *prop list* and return its contents. The implementation code is adapted from the answer provided by Enrico Gregorio (@egreg) in Expand a `.cs` defined by key inside the function.

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

```
5536 \keys_define:nn { enumext / foreach }
5537   {
5538     before  .tl_set:N  = \l__enumext_foreach_before_tl,
5539     before  .value_required:n = true,
5540     after   .tl_set:N  = \l__enumext_foreach_after_tl,
5541     after   .value_required:n = true,
```

```
5542      start    .int_set:N = \l__enumext_foreach_start_int,
5543      start    .value_required:n = true,
5544      stop     .int_set:N = \l__enumext_foreach_stop_int,
5545      stop     .value_required:n = true,
5546      step     .int_set:N = \l__enumext_foreach_step_int,
5547      step     .value_required:n = true,
5548      wrapper .cs_set_protected:Np = \__enumext_foreach_wrapper:n #1,
5549      wrapper .value_required:n = true,
5550      sep      .tl_set:N  = \l__enumext_foreach_sep_tl,
5551      sep      .value_required:n = true,
5552      unknown .code:n    = { \__enumext_parse_foreach_keys:n {#1} }
5553    }
5554 \keys_precompile:nnN { enumext / foreach }
5555    {
5556      before={},after={},start=1,step=1,stop=0,wrapper=#1,sep={; }
5557    }
5558    \l__enumext_foreach_default_keys_tl
```

Functions for handling unknown ⟨keys⟩.

```
5559 \cs_new_protected:Npn \__enumext_parse_foreach_keys:nn #1#2
5560    {
5561      \tl_if_blank:nTF {#2}
5562        {
5563          \msg_error:nnn { enumext } { for-key-unknown } {#1}
5564        }
5565        {
5566          \msg_error:nnnn { enumext } { for-key-value-unknown } {#1} {#2}
5567        }
5568    }
5569 \cs_new_protected:Npn \__enumext_parse_foreach_keys:n #1
5570    {
5571      \exp_args:NV \__enumext_parse_foreach_keys:nn \l_keys_key_str {#1}
5572    }
```

We create the command.

```
5573 \NewDocumentCommand \foreachkeyans { +O{} m }
5574    {
5575      \__enumext_foreach_keyans:nn {#1} {#2}
5576    }
```

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

```
5577 \cs_new_protected:Npn \__enumext_foreach_keyans:nn #1 #2
5578    {
5579      \tl_use:N \l__enumext_foreach_default_keys_tl
5580      \keys_set:nn { enumext / foreach } {#1}
5581      \tl_set:Nn \l__enumext_foreach_name_prop_tl {#2}
5582      \prop_if_exist:cF { g__enumext_#2_prop }
5583        {
5584          \msg_error:nnn { enumext } { undefined-storage-anskey } {#2}
5585        }
5586      \int_compare:nNnT { \l__enumext_foreach_stop_int } = { 0 }
5587        {
5588          \int_set:Nn \l__enumext_foreach_stop_int
5589            { \prop_count:c { g__enumext_#2_prop } }
5590        }
5591      \seq_clear:N \l__enumext_foreach_print_seq
5592      \int_step_function:nnnN
5593        { \l__enumext_foreach_start_int }
5594        { \l__enumext_foreach_step_int }
5595        { \l__enumext_foreach_stop_int }
5596        \__enumext_foreach_add_body:n
5597      \seq_use:NV \l__enumext_foreach_print_seq \l__enumext_foreach_sep_tl
5598    }
5599 \cs_new_protected:Npn \__enumext_foreach_add_body:n #1
5600    {
5601      \seq_put_right:Ne \l__enumext_foreach_print_seq
5602        {
5603          \exp_not:V \l__enumext_foreach_before_tl
5604          \__enumext_foreach_wrapper:n
5605            {
5606              \prop_item:cn { g__enumext_ \l__enumext_foreach_name_prop_tl _prop }{#1}
```

```
5607            }
5608          \exp_not:V \l__enumext_foreach_after_tl
5609        }
5610      }
```

(*End of definition for* \foreachkeyans *and others. This function is documented on page 18.*)

13.51   Messages

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

```
5611 \msg_new:nnn { enumext } { package-load }
5612    {
5613      The ~ '#1' ~ package ~ is ~ already ~ loaded.
5614    }
5615 \msg_new:nnn { enumext } { package-not-load }
5616    {
5617      The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
5618    }
5619 \msg_new:nnn { enumext } { package-load-foot }
5620    {
5621      The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
5622    }
```

Message used in the creation of counters by enumext package.

```
5623 \msg_new:nnn { enumext } { counters }
5624    {
5625      The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
5626      package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
5627    }
```

Message used by align and mark-pos keys.

```
5628 \msg_new:nnn { enumext } { unknown-choice }
5629    {
5630      The ~ value ~ '#3' ~ for ~ '#1' ~ key ~ is ~ invalid ~ use ~ ('#2').
5631    }
```

Message used by reserved anskey* environment by enumext package.

```
5632 \msg_new:nnnn { enumext } { anskey-env-error }
5633    {
5634      The ~ '#1' ~ environment ~is ~ reserved ~ by ~\\
5635      'enumext' ~ package, ~ It~ is~ already~ defined.
5636    }
5637    {
5638      The ~ anskey* ~ environment ~ is ~ defined ~ internally ~
5639      for ~ the ~ 'save-ans' ~ key.\\
5640    }
```

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

```
5641 \msg_new:nnn { enumext } { store-prop }
5642    {
5643      * ~ Package ~ enumext: ~ Creating ~
5644       \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
5645    }
5646 \msg_new:nnn { enumext } { store-seq }
5647    {
5648      * ~ Package ~ enumext: ~ Creating ~
5649      \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
5650    }
5651 \msg_new:nnn { enumext } { store-int }
5652    {
5653      * ~ Package ~ enumext: ~ Creating ~
5654      \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_context:.
5655    }
5656 \msg_new:nnn { enumext } { prop-seq-int-hook }
5657    {
5658      * ~ Package ~ enumext: ~ Elements ~ in ~
5659      \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
5660      * ~ Package ~ enumext: ~ Elements ~ in ~
5661      \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
5662      * ~ Package ~ enumext: ~ Value ~ off ~
5663      \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
5664    }
5665 \msg_new:nnn { enumext } { item-answer-hook }
```

```
5666    {
5667      * ~ Package ~ enumext: ~ Value ~ off ~
5668      \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\\
5669      * ~ Package ~ enumext: ~ Value ~ off ~
5670      \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\\
5671      * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
5672    }
```

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

```
5673  \msg_new:nnn { enumext } { invalid-key }
5674    {
5675      The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
5676    }
5677  \msg_new:nnn { enumext } { unknown-key-family }
5678    {
5679      Unknown~key~family~`\l_keys_key_str'~for~enumext.
5680    }
```

Messages used in length calculation.

```
5681  \msg_new:nnn { enumext } { width-negative }
5682    {
5683      Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\\
5684      The ~ key ~ '#1'~ accepts ~ values  ~ >= ~ 0pt.
5685    }
5686  \msg_new:nnn { enumext } { width-zero }
5687    {
5688      Invalid ~ '#1=#2' ~ \msg_line_context:.\\
5689      The ~ key ~ '#1'~ accepts ~ values  ~ > ~ 0pt.
5690    }
```

Messages used by show-length key in enumext.

```
5691  \msg_new:nnn { enumext } { list-lengths }
5692    {
5693      **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
5694      \__enumext_show_length:nnn { dim  } { labelsep      } {#1}
5695      \__enumext_show_length:nnn { dim  } { labelwidth    } {#1}
5696      \__enumext_show_length:nnn { dim  } { itemindent    } {#1}
5697      \__enumext_show_length:nnn { dim  } { leftmargin    } {#1}
5698      \__enumext_show_length:nnn { dim  } { rightmargin   } {#1}
5699      \__enumext_show_length:nnn { dim  } { listparindent } {#1}
5700      \__enumext_show_length:nnn { skip } { topsep      } {#1}
5701      \__enumext_show_length:nnn { skip } { parsep      } {#1}
5702      \__enumext_show_length:nnn { skip } { partopsep } {#1}
5703      \__enumext_show_length:nnn { skip } { itemsep    } {#1}
5704      **************************************************
5705    }
```

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

```
5706  \msg_new:nnn { enumext } { list-lengths-not-nested }
5707    {
5708      **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
5709      \__enumext_show_length:nnn { dim  } { labelsep      } {#1}
5710      \__enumext_show_length:nnn { dim  } { labelwidth    } {#1}
5711      \__enumext_show_length:nnn { dim  } { itemindent    } {#1}
5712      \__enumext_show_length:nnn { dim  } { leftmargin    } {#1}
5713      \__enumext_show_length:nnn { dim  } { rightmargin   } {#1}
5714      \__enumext_show_length:nnn { dim  } { listparindent } {#1}
5715      \__enumext_show_length:nnn { skip } { topsep      } {#1}
5716      \__enumext_show_length:nnn { skip } { parsep      } {#1}
5717      \__enumext_show_length:nnn { skip } { partopsep } {#1}
5718      \__enumext_show_length:nnn { skip } { itemsep    } {#1}
5719      **************************************************
5720    }
```

Messages used by ref key.

```
5721  \msg_new:nnn { enumext } { key-ref-empty }
5722    {
5723      Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
5724    }
```

Messages used by save-ans key.

```
5725  \msg_new:nnn { enumext } { save-ans-empty }
5726    {
5727      Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
```

```
5728        }
5729  \msg_new:nnn { enumext } { save-ans-log }
5730        {
5731          * ~ Package ~ enumext: ~ Start ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5732        }
5733  \msg_new:nnn { enumext } { save-ans-log-hook }
5734        {
5735          * ~ Package ~ enumext: ~ Stop ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5736        }
5737  \msg_new:nnn { enumext } { save-ans-hook }
5738        {
5739          Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
5740        }
```

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

```
5741  \msg_new:nnn { enumext } { need-save-ans }
5742        {
5743          Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
5744        }
5745  \msg_new:nnn { enumext } { items-same-answer }
5746        {
5747          ****************************************\\
5748          * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~
5749          for ~ \c_left_brace_str #2 \c_right_brace_str\\
5750          * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~
5751          'OK', ~ all ~ items ~ with ~ answer.\\
5752          ****************************************
5753        }
5754  \msg_new:nnn { enumext } { item-greater-answer }
5755        {
5756          Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
5757          started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
5758          Items ~ > ~ Answers.
5759        }
5760  \msg_new:nnn { enumext } { item-less-answer }
5761        {
5762          Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
5763          started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
5764          Items ~ < ~ Answers.
5765        }
```

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

```
5766  \msg_new:nnn { enumext } { missing-starred }
5767        {
5768          Missing ~ '\c_backslash_str #1*' ~ #2.
5769        }
5770  \msg_new:nnn { enumext } { many-starred }
5771        {
5772          Many ~ '\c_backslash_str #1*' ~ #2.
5773        }
```

Messages used by \printkeyans* command.

```
5774  \msg_new:nnn { enumext } { print-starred }
5775        {
5776          \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
5777          #2 ~ environment ~  \msg_line_context:.
5778        }
```

Message for the nesting depth of the environment enumext.

```
5779  \msg_new:nnn { enumext } { list-too-deep }
5780        {
5781          Too ~ deep ~ nesting  ~ for ~ 'enumext' ~ \msg_line_context:.~ \\
5782          The ~ maximum  ~ level  ~ of  ~ nesting  ~ is ~ 4.
5783        }
```

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

```
5784  \msg_new:nnn { enumext } { anskey-unnumber-item }
5785        {
5786          Can't ~ store ~ with ~ a ~ unnumbered ~ \c_backslash_str item ~ \msg_line_context:.
5787        }
5788  \msg_new:nnn { enumext } { anskey-already-stored }
5789        {
5790          Content ~ already ~ stored ~ for ~ this ~ \c_backslash_str item ~ \msg_line_context:.
```

```
5791      }
5792  \msg_new:nnn { enumext } { anskey-empty-arg }
5793      {
5794        Can't ~ store ~ empty ~ content ~ \msg_line_context:.
5795      }
5796  \msg_new:nnn { enumext } { anskey-wrong-place }
5797      {
5798        Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
5799        '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5800      }
5801  \msg_new:nnn { enumext } { anskey-nested }
5802      {
5803        The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
5804      }
5805  \msg_new:nnn { enumext } { anskey-math-mode }
5806      {
5807        #1 ~ can't ~ work ~ in ~ math ~ mode ~ \msg_line_context:.
5808      }
5809  \msg_new:nnn { enumext } { anskey-env-wrong }
5810      {
5811        The ~ environment ~ anskey* ~ cannot ~ use ~ in ~ '#1' ~ \msg_line_context:.
5812      }
5813  \msg_new:nnn { enumext } { anspic-wrong-place }
5814      {
5815        Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
5816        '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5817      }
5818  \msg_new:nnn { enumext } { command-wrong-place }
5819      {
5820        Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
5821        '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
5822      }
5823  \msg_new:nnnn { enumext } { anskey-env-key-unknown }
5824      {
5825        The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5826        'anskey*' ~ and ~ is ~ being ~ ignored.
5827      }
5828      {
5829        The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5830        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5831      }
5832  \msg_new:nnnn { enumext } { anskey-env-key-value-unknown }
5833      {
5834        The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5835        'anskey*' ~ and ~ is ~ being ~ ignored.
5836      }
5837      {
5838        The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5839        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5840      }
5841  \msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
5842      { The ~ key ~'#1'~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.}
5843      {
5844        The ~ command ~'\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5845        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5846      }
5847  \msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
5848      { The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignor
5849      {
5850        The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5851        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5852      }
```

Messages used by keyans, keyans* and keyanspic environment.

```
5853  \msg_new:nnn { enumext } { keyans-nested }
5854      {
5855        The ~ environment ~ 'keyans' ~ can't ~ be  ~ nested  ~ \msg_line_context:.
5856      }
5857  \msg_new:nnn { enumext } { keyans-wrong-level }
5858      {
5859        Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:.~ \\
5860        The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
```

```
5861      }
5862 \msg_new:nnn { enumext } { wrong-place }
5863      {
5864        Wrong ~ place ~ for ~ '#1' ~ environment ~\msg_line_context:.~ \\
5865        '#1' ~ is ~ only ~ found ~ with ~ '#2' ~  in  ~  'enumext.
5866      }
5867 \msg_new:nnn { enumext } { keyanspic-nested }
5868      {
5869        The ~ environment ~ 'keyanspic' ~ can't ~ be  ~ nested~ \msg_line_context:.~.
5870      }
5871 \msg_new:nnn { enumext } { keyanspic-wrong-level }
5872      {
5873        Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:.~ \\
5874        The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5875      }
5876 \msg_new:nnn { enumext } { keyanspic-item-cmd }
5877      {
5878        Can't ~ use  ~ \c_backslash_str item ~ in ~ keyanspic ~ \msg_line_context:.
5879      }
5880 \msg_new:nnnn { enumext } { keyans-unknown-key }
5881      {
5882        The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5883        '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5884      }
5885      {
5886        The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5887        ~ have ~ a ~ key ~ called ~'#1'.\\
5888        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5889      }
5890 \msg_new:nnnn { enumext } { keyans-unknown-key-value }
5891      {
5892        The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5893        '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5894      }
5895      {
5896        The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5897        ~ have ~ a ~ key ~ called ~'#1'.\\
5898        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5899      }
```

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

```
5900 \msg_new:nnnn { enumext } { starred-unknown-key }
5901      {
5902        The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5903        '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5904      }
5905      {
5906        The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5907        ~ have ~ a ~ key ~ called ~'#1'.\\
5908        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5909      }
5910 \msg_new:nnnn { enumext } { starred-unknown-key-value }
5911      {
5912        The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5913        '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5914      }
5915      {
5916        The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5917        ~ have ~ a ~ key ~ called ~'#1'.\\
5918        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5919      }
```

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

```
5920 \msg_new:nnnn { enumext } { standar-unknown-key }
5921      {
5922        The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_tl
5923        ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5924      }
5925      {
5926        The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5927        ~ have ~ a ~ key ~ called ~'#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5928        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
```

```
5929        }
5930   \msg_new:nnnn { enumext } { standar-unknown-key-value }
5931        {
5932          The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_
5933          ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5934        }
5935        {
5936          The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5937          ~ have ~ a ~ key ~ called ~'#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5938          Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5939        }
```

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

```
5940   \msg_new:nnnn { enumext } { for-key-unknown }
5941        { The~key~'#1'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored.}
5942        {
5943          The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5944          Check~that~you~have~spelled~the~key~name~correctly.
5945        }
5946   \msg_new:nnnn { enumext } { for-key-value-unknown }
5947        { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored. }
5948        {
5949          The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5950          Check~that~you~have~spelled~the~key~name~correctly.
5951        }
```

Messages used by \getkeyans command.

```
5952   \msg_new:nnn { enumext } { undefined-storage-anskey }
5953        {
5954          Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
5955        }
```

Messages used by \miniright command.

```
5956   \msg_new:nnn { enumext } { missing-miniright }
5957        {
5958          Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\\
5959          The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
5960        }
5961   \msg_new:nnn { enumext } { wrong-miniright-place }
5962        {
5963          Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
5964          Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
5965        }
5966   \msg_new:nnn { enumext } { wrong-miniright-use }
5967        {
5968          Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
5969          '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
5970        }
5971   \msg_new:nnn { enumext } { wrong-miniright-starred }
5972        {
5973          Can't ~ use  ~ \c_backslash_str miniright ~ in ~ starred ~ environments ~ \msg_line_context:.
5974        }
5975   \msg_new:nnn { enumext } { many-miniright-used }
5976        {
5977          Can't ~ use  ~ \c_backslash_str miniright ~ more ~ than ~ once ~  \msg_line_context:.
5978        }
```

Messages used by \setenumextmeta command.

```
5979   \msg_new:nnn { enumext } { unknown-set }
5980        {
5981          Argument ~ [#1] ~ is ~ unknown ~ by ~  \c_backslash_str setenumextmeta ~ \msg_line_context:.
5982        }
5983   \msg_new:nnn { enumext } { already-defined }
5984        {
5985          The ~ key ~ '#1' ~ is ~ already ~ defined ~ \msg_line_context:.
5986        }
5987   \msg_new:nnn { enumext } { prohibited-unknown }
5988        {
5989          The ~ name ~ 'unknown' ~ can't ~ be ~ chosen~ for ~ a ~ meta ~ key ~ \msg_line_context:.
5990        }
```

Messages used by enumext* and keyans* environments.

```
5991   \msg_new:nnn { enumext } { nested }
```

```
5992    {
5993      The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ \msg_line_con
5994    }
5995  \msg_new:nnn { enumext } { nested-horizontal }
5996    {
5997      The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ in ~ '#1' ~ '
5998    }
5999  \msg_new:nnn { enumext } { item-joined }
6000    {
6001      Items ~ joined ~ (#1) ~ > ~ #2  ~ columns ~\msg_line_context:.
6002    }
6003  \msg_new:nnn { enumext } { item-joined-columns }
6004    {
6005      Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.
6006    }
```

13.52 Finish package

Finish package implementation.

```
6007  \file_input_stop:
6008  ⟨/package⟩
```

14 Index of Implementation

The italic numbers denote the pages where the corresponding entry is described, the numbers underlined and all others indicate the line on which they are implemented in the package code.