enumext
ENUMERATE EXERCISE SHEETS

v1.3   2025-06-01[*]

©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` package.

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] package, 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 2025-06-01.

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*.

<div>

enumext
enumext*

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

</div>

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*`

<div>

`\item*`

`\item*[`⟨*symbol*⟩`][`⟨*offset*⟩`]`

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

The *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).

</div>

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

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

The command \setenumext sets the ⟨keys⟩ on a global basis for environments enumext, enumext*, keyans, keyans* and the \printkeys 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 enumext 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 argument c *"collect code"* implemented in the ltcmd module, part of LaTeX release 2025-06-01. \begin{anskey*} and \end{anskey*} must be in different lines and should not appear within verbatim environments or commands. 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, LaTeX will return an *"error"* message if this happens. In a similar way comments *"%"* or *"any character"* after \end{anskey*} on the same line LaTeX 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*⟩ write-env, overwrite and force-eol. 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}`%.

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.　　7. Text containing our instructions or questions.

[5] | First answer with `verbatim` |　　[7] | third answer |

6. Text containing our instructions or questions.　　8. Text containing our instructions or questions.

(a) Question.　　[8] | fourth answer |

[6] | second 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
   - ∗ B) Correct choice
   - C) Choice
   - D) Choice
   - E) Choice

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

   Some text

6.5 The environment keyanspic

keyanspic
```
\begin{keyanspic}[⟨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.
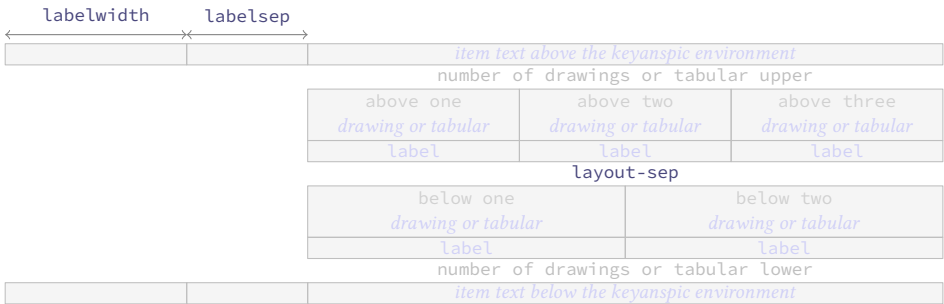
Figure 6: Representation of the keyanspic environment with layout-sty={⟨3, 2⟩} in enumext.

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)
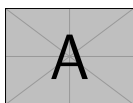
D)

∗ E) [note]

2. Question with images and labels above.
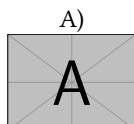
A)
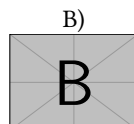
B)
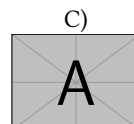
C)

D)
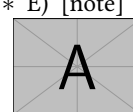
∗ E) [note]

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:

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

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

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

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

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

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

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

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

Example 2

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

---

[1] 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
   - Ⓒ value
   - Ⓑ correct
   - Ⓓ value

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

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

4. Question with image and label below:

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

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) $\LaTeX$2e 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.

2

Which choice best describes what happens in the passage?

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

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

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

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

3

Which choice best describes what happens in the passage?

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

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

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

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

4

Which choice best describes what happens in the passage?

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

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

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

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

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

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) 12. B (10.5a, KSC)
11. D (9.5a, 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, 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 LoNg A much longer description. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris.

Final notes

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

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

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

Why list environments?

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

Why not random questions and other utilities

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

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, 2025.

[4] González, Pablo. "scontents - Stores LaTeX contents in memory or files". Available from CTAN, https://www.ctan.org/pkg/scontents, 2025.

[5] The LaTeX Project. "enumerate – Enumerate with redefinable labels". Available from CTAN, https://www.ctan.org/pkg/enumerate, 2025.

[6] Bezos, Javier. "Customizing lists with the enumitem package". Available from CTAN, https://www.ctan.org/pkg/enumitem, 2025.

[7] Berry, Karl. "LaTeX 2ε: An Unofficial Reference Manual". Available from CTAN, https://ctan.org/pkg/latex2e-help-texinfo, 2025.

[8] The LaTeX Project. "Extensive support for hypertext in LaTeX". Available from CTAN, https://www.ctan.org/pkg/hyperref, 2025.

[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, 2025.

[11] The LaTeX Project. "The LaTeX3 Interfaces". Available from CTAN, https://www.ctan.org/pkg/l3kernel, 2025.

[12] The LaTeX Project. "The LaTeX 2ε sources". Available from CTAN, https://ctan.org/tex-archive/macros/latex/base, 2025.

[13] The LaTeX Project. "LaTeX for authors current version". Available from CTAN, https://ctan.org/pkg/latex-base, 2025.

[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, 2025.

[18] The LaTeX Project. "latex-lab – LaTeX laboratory". Available from CTAN, https://www.ctan.org/pkg/latex-lab, 2025.

[19] MITTELBACH, FRANK. "LaTeX's socket management". Available from CTAN, https://ctan.org/tex-archive/macros/latex/base, 2025.

11   Change history

v1.3 (ctan), 2025-06-01	– Removed dependency on the `scontents` package.
	– The `anskey*` environment has been rewritten using the new c argument from `ltcmd` module.
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}[2025-06-01]
```

Now declare the enumext package.

```
4 \ProvidesExplPackage {enumext} {2025-06-01} {1.3} {Enumerate exercise sheets}
```

Finally check if the `multicol` package 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    }
```

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.

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

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

```
16 \int_new:N  \l__enumext_level_int
17 \int_new:N  \l__enumext_level_h_int
18 \int_new:N  \l__enumext_anskey_level_int
19 \int_new:N  \l__enumext_keyans_level_int
20 \int_new:N  \l__enumext_keyans_level_h_int
21 \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_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).

```
22 \bool_new:N \l__enumext_starred_bool
23 \bool_new:N \g__enumext_starred_bool
24 \bool_new:N \l__enumext_starred_first_bool
25 \bool_new:N \l__enumext_standar_bool
26 \bool_new:N \g__enumext_standar_bool
27 \bool_new:N \l__enumext_standar_first_bool
28 \bool_new:N \l__enumext_keyans_env_bool
29 \tl_new:N   \g__enumext_start_line_tl
30 \tl_new:N   \g__enumext_envir_name_tl
31 \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).

```
32 \cs_set_protected:Npn \__enumext_tmp:n #1
33   {
34     \tl_new:c { l__enumext_counter_#1_tl }
35   }
36 \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).

```
37 \tl_const:Nn \c__enumext_counter_style_tl
38   { { arabic } { roman } { Roman } { alph } { Alph } }
39 \tl_new:N \l__enumext_ref_key_arg_tl
40 \tl_new:N \l__enumext_ref_the_count_tl
41 \cs_set_protected:Npn \__enumext_tmp:n #1
42   {
43     \tl_new:c  { l__enumext_renew_the_count_#1_tl }
44     \tl_new:c  { l__enumext_the_counter_#1_tl }
45     \tl_set:ce { l__enumext_the_counter_#1_tl } { \exp_not:c { theenumX#1 } }
46   }
47 \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).

```
48 \int_new:N  \g__enumext_resume_int
49 \int_new:N  \g__enumext_resume_vii_int
50 \tl_new:N   \l__enumext_resume_name_tl
51 \bool_new:N \l__enumext_resume_active_bool
52 \tl_new:N   \g__enumext_standar_series_tl
53 \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.

```
54 \dim_new:N \l__enumext_current_widest_dim
55 \tl_new:N  \g__enumext_counter_styles_tl
56 \tl_new:N  \g__enumext_widest_label_tl
57 \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).

```
58 \cs_set_protected:Npn \__enumext_tmp:n #1
59   {
60     \bool_new:c { l__enumext_leftmargin_tmp_#1_bool }
```

```
61        \dim_new:c  { l__enumext_leftmargin_tmp_#1_dim  }
62        \dim_new:c  { l__enumext_leftmargin_#1_dim        }
63        \dim_new:c  { l__enumext_itemindent_#1_dim        }
64     }
65  \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).

```
66  \cs_set_protected:Npn \__enumext_tmp:n #1
67     {
68        \skip_new:c  { l__enumext_multicols_above_#1_skip }
69        \skip_new:c  { l__enumext_multicols_below_#1_skip }
70        \skip_new:c  { g__enumext_multicols_right_#1_skip }
71        \str_new:c   { l__enumext_align_label_pos_#1_str  }
72     }
73  \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).

```
74  \int_new:N  \g__enumext_minipage_stat_int
75  \skip_new:N \l__enumext_minipage_temp_skip
76  \skip_new:N \l__enumext_minipage_left_skip
77  \skip_new:N \l__enumext_minipage_right_skip
78  \skip_new:N \l__enumext_minipage_after_skip
79  \skip_new:N \g__enumext_minipage_right_skip
80  \skip_new:N \g__enumext_minipage_after_skip
81  \cs_set_protected:Npn \__enumext_tmp:n #1
82     {
83        \dim_new:c  { l__enumext_minipage_left_#1_dim    }
84        \bool_new:c { l__enumext_minipage_active_#1_bool }
85     }
86  \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).

```
87  \cs_set_protected:Npn \__enumext_tmp:n #1
88     {
89        \bool_new:c { l__enumext_wrap_label_#1_bool      }
90        \bool_new:c { l__enumext_wrap_label_opt_#1_bool }
91        \int_new:c  { l__enumext_start_#1_int             }
92        \tl_new:c   { l__enumext_fake_item_indent_#1_tl }
93        \tl_new:c   { l__enumext_label_fill_left_#1_tl  }
94        \tl_new:c   { l__enumext_label_fill_right_#1_tl }
95        \bool_new:c { l__enumext_vspace_a_star_#1_bool  }
96        \bool_new:c { l__enumext_vspace_b_star_#1_bool  }
97     }
98  \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }
```

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

`\l__enumext_store_active_bool`
`\l__enumext_store_name_tl`
`\g__enumext_store_name_tl`
`\l__enumext_store_current_label_tl`
`\l__enumext_store_current_opt_arg_tl`
`\l__enumext_store_current_label_tmp_tl`

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

```
 99 \bool_new:N \l__enumext_store_active_bool
100 \tl_new:N   \l__enumext_store_name_tl
101 \tl_new:N   \g_enumext_store_name_tl
102 \tl_new:N   \l__enumext_store_current_label_tl
103 \tl_new:N   \l__enumext_store_current_opt_arg_tl
104 \tl_new:N   \l__enumext_store_current_label_tmp_tl
```

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

`\l__enumext_store_anskey_arg_tl`
`\l__enumext_store_anskey_env_tl`
`\l__enumext_write_anskey_env_bool`
`\l__enumext_write_anskey_env_file_name_tl`
`\l__enumext_write_anskey_env_file_iow`

The variable `\l__enumext_store_anskey_arg_tl` save the *argument* of `\anskey` (§13.30) and the variables `\l__enumext_store_anskey_env_tl` save the ⟨*body*⟩ of the environment `anskey*` (§13.31).

The variables `\l__enumext_write_anskey_env_bool`, `\l__enumext_write_anskey_env_file_name_-tl` and `\l__enumext_write_anskey_env_file_iow` they are used by the `write-env` and `overwrite` keys in the `anskey*` environment implementation.

```
105 \tl_new:N   \l__enumext_store_anskey_arg_tl
106 \tl_new:N   \l__enumext_store_anskey_env_tl
107 \bool_new:N \l__enumext_write_anskey_env_bool
108 \tl_new:N   \l__enumext_write_anskey_env_file_name_tl
109 \iow_new:N  \l__enumext_write_anskey_env_file_iow
```

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

`\c__enumext_anskey_env_hidden_space_str`

The `\c__enumext_anskey_env_hidden_space_str` is a constant *string* to used to hide the ⟨*forced space*⟩ added by TeX when recording content in a macro. This *string* contains the *reserved phrase* "`%^^Aenumextheol%`" which is added to the end of the argument stored in *sequence* and *prop list* when the key `force-eol` is false.

```
110 \str_const:Ne \c__enumext_anskey_env_hidden_space_str
111   { \c_percent_str \c_circumflex_str \c_circumflex_str A enumextheol \c_percent_str }
```

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

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

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

```
112 \tl_new:N  \l__enumext_setkey_tmpa_tl
113 \tl_new:N  \l__enumext_setkey_tmpb_tl
114 \int_new:N \l__enumext_setkey_tmpa_int
115 \seq_new:N \l__enumext_setkey_tmpa_seq
116 \seq_new:N \l__enumext_setkey_tmpb_seq
```

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

`\l__enumext_meta_path_tl`
`\l__enumext_foreach_print_seq`
`\l__enumext_foreach_name_prop_tl`
`\l__enumext_foreach_default_keys_tl`

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

```
117 \tl_new:N  \l__enumext_meta_path_tl
118 \seq_new:N \l__enumext_foreach_print_seq
119 \tl_new:N  \l__enumext_foreach_name_prop_tl
120 \tl_new:N  \l__enumext_foreach_default_keys_tl
```

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

`\l__enumext_print_keyans_starred_tl`
`\l__enumext_print_keyans_star_bool`
`\l__enumext_mark_position_str`
`\l__enumext_mark_position_v_str`
`\l__enumext_mark_position_viii_str`
`\l__enumext_mark_sep_tmpa_dim`
`\l__enumext_mark_sep_tmpb_dim`
`\l__enumext_show_pos_tmp_int`
`\g__enumext_item_symbol_aux_tl`
`\l__enumext_print_keyans_X_tl`
`\l__enumext_store_save_key_X_tl`
`\l__enumext_store_save_key_X_bool`
`\l__enumext_store_upper_level_X_bool`

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

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

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

\l__enumext_anspic_args_seq

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

```
138 \seq_new:N  \l__enumext_anspic_args_seq
139 \dim_new:N  \l__enumext_anspic_mini_width_dim
140 \int_new:N  \l__enumext_anspic_above_int
141 \int_new:N  \l__enumext_anspic_below_int
142 \bool_new:N \l__enumext_anspic_label_above_bool
143 \str_new:N  \l__enumext_anspic_mini_pos_str
144 \box_new:N  \l__enumext_anspic_label_box
145 \box_new:N  \l__enumext_anspic_body_box
146 \dim_new:N  \l__enumext_anspic_label_htdp_dim
147 \dim_new:N  \l__enumext_anspic_body_htdp_dim
```

\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

(*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.

```
148 \bool_new:N \l__enumext_check_answers_bool
149 \bool_new:N \g__enumext_check_ans_key_bool
150 \tl_new:N    \l__enumext_check_start_line_env_tl
151 \bool_new:N \l__enumext_item_wrap_key_bool
152 \int_new:N  \g__enumext_check_starred_cmd_int
153 \int_new:N  \g__enumext_item_anskey_int
154 \int_new:N  \g__enumext_item_number_int
155 \bool_new:N \l__enumext_item_number_bool
156 \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`.

```
157 \bool_new:N \l__enumext_hyperref_bool
158 \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.

```
159 \tl_new:N \l__enumext_newlabel_arg_one_tl
160 \tl_new:N \l__enumext_newlabel_arg_two_tl
161 \tl_new:N \l__enumext_write_aux_file_tl
162 \cs_set_protected:Npn \__enumext_tmp:n #1
163   {
164     \tl_new:c { l__enumext_label_copy_#1_tl }
165   }
166 \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).

```
167 \int_new:N \g__enumext_footnote_standar_int
168 \int_new:N \g__enumext_footnote_starred_int
169 \seq_new:N \g__enumext_footnote_standar_arg_seq
170 \seq_new:N \g__enumext_footnote_starred_arg_seq
171 \seq_new:N \g__enumext_footnote_standar_int_seq
172 \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.

```
173 \cs_set_protected:Npn \__enumext_tmp:n #1
174   {
175     \bool_new:c { l__enumext_item_starred_#1_bool     }
176     \int_new:c  { l__enumext_item_column_pos_#1_int   }
177     \int_new:c  { g__enumext_item_count_all_#1_int    }
178     \int_new:c  { l__enumext_joined_item_#1_int       }
```

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

```
194  \clist_const:Nn \c__enumext_all_envs_clist
195    {
196      {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
197      {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
198    }
```

(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).

```
199  \cs_generate_variant:Nn \keys_precompile:nnN { neN }
200  \cs_generate_variant:Nn \seq_use:Nn { NV }
```

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

`\__enumext_scan_tokens:n` The functions `\tl_rescan:nn` and `\tl_set_rescan:Nnn` provided by expl3 doesn't fit the needs of this package because it does not allow catcode changes inside the argument, so verbatim stuff used inside one of `anskey*` environment will not work. Here we create a private copy of `\tex_scantokens:D` which will serve our purposes. See the answer by Ulrich Diez in How do use {<setup>} in \tl_set_rescan:Nnn to replace \scantokens?.

```
201  \cs_new_protected:Npn \__enumext_scan_tokens:n #1 { \tex_scantokens:D {#1} }
```

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

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

```
202  \cs_new_protected:Npn \__enumext_at_begin_document:n #1
203    {
204      \hook_gput_code:nnn {begindocument} {enumext} { #1 }
205    }
```

(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.

```
206  \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
207    {
208      \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
209    }
210  \cs_new_protected:Npn \__enumext_before_env:nn #1 #2
211    {
212      \hook_gput_code:nnn {env/#1/before} {enumext} {#2}
213    }
```

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

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

```
214  \cs_new:Nn \__enumext_level:
215    {
216      \int_to_roman:n { \l__enumext_level_int }
217    }
```

(*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?.

```
218  \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
219    {
220      \regex_match:nnTF { ^[\+\-]?[\d]+$ } {#1} % $
221        { \prg_return_true: }
222        { \prg_return_false: }
223    }
```

(*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.

```
224  \cs_new_protected:Nn \__enumext_regex_counter_style:
225    {
226      \tl_map_inline:Nn \c__enumext_counter_style_tl
227        {
228          \regex_replace_once:nnN { \c{##1}\* }
229            { \c{##1}\cB{\u{l__enumext_ref_the_count_tl}\cE} } \l__enumext_ref_key_arg_tl
230        }
231    }
```

(*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.

```
232  \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
233    {
234      *~#2
235      \prg_replicate:nn { 14 - \str_count:n {#2} } {~}
236        =~\use:c { #1_use:c } { l__enumext_#2_#3_#1 } \\
237    }
```

(*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.

```
238  \cs_new_protected:Nn \__enumext_unskip_unkern:
239    {
240      \int_case:nnT { \lastnodetype }
241        {
242          { 11 }{ \unskip }
243          { 12 }{ \unkern }
244        }
245    }
```

(*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`.

```
246  \cs_new_protected:Nn \__enumext_is_not_nested:
247    {
248      \str_case:en { \@currenvir }
249        {
250          {enumext}
251            {
252              \tl_set:Nn \l__enumext_envir_name_tl { enumext }
253              \bool_lazy_and:nnT
254                { \bool_not_p:n { \g__enumext_standar_bool } }
255                { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
256                {
257                  \bool_gset_true:N \g__enumext_standar_bool
258                }
```

```
259              }
260          {enumext*}
261              {
262                \tl_set:Nn \l__enumext_envir_name_tl { enumext* }
263                \bool_lazy_and:nnT
264                  { \bool_not_p:n { \g__enumext_starred_bool } }
265                  { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
266                  {
267                    \bool_gset_true:N \g__enumext_starred_bool
268                  }
269              }
270          }
271      }
```

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) 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.

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

(*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.

```
303 \cs_new_protected:Nn \__enumext_keyans_name_and_start:
304     {
305        \str_case:en { \@currenvir }
306            {
307                {keyans}
308                    {
309                        \tl_set:Nn \l__enumext_envir_name_tl { keyans }
310                        \tl_set:Ne \l__enumext_check_start_line_env_tl
311                            {
312                                in~'keyans'~start~on~line~\exp_not:V \inputlineno
313                            }
314                    }
315                {keyans*}
316                    {
317                        \tl_set:Nn \l__enumext_envir_name_tl { keyans* }
```

```
318              \tl_set:Ne \l__enumext_check_start_line_env_tl
319                {
320                  in~'keyans*'~start~on~line~\exp_not:V \inputlineno
321                }
322            }
323         {keyanspic}
324            {
325              \tl_set:Nn \l__enumext_envir_name_tl { keyanspic }
326              \tl_set:Ne \l__enumext_check_start_line_env_tl
327                {
328                  in~'keyanspic'~start~on~line~\exp_not:V \inputlineno
329                }
330            }
331        }
332     }
```

(*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.

```
333 \cs_new_protected:Nn \__enumext_reset_global_vars:
334   {
335     \__enumext_reset_global_int:
336     \__enumext_reset_global_bool:
337     \__enumext_reset_global_tl:
338   }
339 \cs_new_protected:Nn \__enumext_reset_global_int:
340   {
341     \int_gzero:N \g__enumext_item_number_int
342     \int_gzero:N \g__enumext_item_anskey_int
343     \int_gzero:N \g__enumext_item_answer_diff_int
344   }
345 \cs_new_protected:Nn \__enumext_reset_global_bool:
346   {
347     \bool_gset_false:N \g__enumext_check_ans_key_bool
348     \bool_gset_false:N \g__enumext_standar_bool
349     \bool_gset_false:N \g__enumext_starred_bool
350   }
351 \cs_new_protected:Nn \__enumext_reset_global_tl:
352   {
353     \tl_gclear:N \g__enumext_store_name_tl
354     \tl_gclear:N \g__enumext_start_line_tl
355     \tl_gclear:N \g__enumext_envir_name_tl
356   }
```

(*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.

```
357 \cs_new_protected:Nn \__enumext_log_global_vars:
358   {
359     \msg_log:nneeee { enumext } { prop-seq-int-hook }
360       { \g__enumext_store_name_tl }
361       { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
362       { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
363       { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
364   }
```

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.

```
365 \cs_new_protected:Nn \__enumext_log_answer_vars:
366   {
367     \msg_log:nneee { enumext } { item-answer-hook }
368       { \int_use:N \g__enumext_item_number_int }
369       { \int_use:N \g__enumext_item_anskey_int }
370       { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int} }
371   }
```

(*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.

```
372 \__enumext_at_begin_document:n
373   {
374     \cs_new_eq:NN    \__enumext_start_list:nn \list
375     \cs_new_eq:NN    \__enumext_stop_list: \endlist
376     \NewCommandCopy \__enumext_item_std:w \item
377   }
```

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

```
378 \__enumext_at_begin_document:n
379   {
380     \cs_new_eq:NN \__enumext_minipage:w \minipage
381     \cs_new_eq:NN \__enumext_endminipage: \endminipage
382   }
```

(*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.

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

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

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`.

```
398     \bool_if:NT \l__enumext_footnotes_key_bool
399       {
400         \IfPackageLoadedTF { footnotehyper }
401           {
402             \msg_info:nnn { enumext } { package-load } { footnotehyper }
403           }
404           {
```

```
405        \bool_set_false:N \l__enumext_footnotes_key_bool
406      }
407    }
```

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.

```
408    \bool_if:NTF \l__enumext_hyperref_bool
409      {
410        \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
411        \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
412      }
413      {
414        \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
415        \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
416      }
417    }
```

(*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.

```
418  \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
419    {
420      \protected@write \@auxout { }
421        {
422          \token_to_str:N \newlabel {#1}
423            {
424              {#2}
425              \bool_if:NT \l__enumext_hyperref_bool
426                { { \thepage } {#2} {#1} }
427              { }
428            }
429        }
430      \__enumext_hypertarget:nn {#1} { }
431      \__enumext_phantomsection:
432    }
```

(*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
__enumext_renew_footnote:
__enumext_print_footnote:
__enumext_renew_footnote_mini:
__enumext_print_footnote_mini:

Redefinition of the \footnote command using \footnotetext and \footnotemark for the mini-env key in the enumext and keyans environments.

```
433  \cs_new_protected:Nn \__enumext_footnotetext:nn
434    {
435      \footnotetext[#1]{#2}
436    }
437  \cs_new_protected:Nn \__enumext_renew_footnote:
438    {
439      \RenewDocumentCommand \footnote { o +m }
440        {
441          \tl_if_novalue:nTF {##1}
442            {
443              \stepcounter{footnote}
444              \int_gset_eq:Nc \g__enumext_footnote_standar_int { c@footnote }
445            }
446            {
447              \int_gset:Nn \g__enumext_footnote_standar_int { ##1 }
448            }
449          \footnotemark [ \g__enumext_footnote_standar_int ]
450          \seq_gput_right:Nn \g__enumext_footnote_standar_arg_seq { ##2 }
451          \seq_gput_right:NV
```

```
452              \g__enumext_footnote_standar_int_seq \g__enumext_footnote_standar_int
453          }
454      }
455  \cs_new_protected:Nn \__enumext_print_footnote:
456      {
457      \seq_if_empty:NF \g__enumext_footnote_standar_int_seq
458          {
459          \seq_map_pairwise_function:NNN
460              \g__enumext_footnote_standar_int_seq
461              \g__enumext_footnote_standar_arg_seq
462              \__enumext_footnotetext:nn
463          }
464      \seq_gclear:N \g__enumext_footnote_standar_arg_seq
465      \seq_gclear:N \g__enumext_footnote_standar_int_seq
466      }
```

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.

```
467  \cs_new_protected:Nn \__enumext_renew_footnote_mini:
468      {
469      \RenewDocumentCommand \footnote { o +m }
470          {
471          \tl_if_novalue:nTF {##1}
472              {
473              \stepcounter{footnote}
474              \int_gset_eq:Nc \g__enumext_footnote_starred_int { c@footnote }
475              }
476              {
477              \int_gset:Nn \g__enumext_footnote_starred_int { ##1 }
478              }
479          \footnotemark [ \g__enumext_footnote_starred_int ]
480          \seq_gput_right:Nn \g__enumext_footnote_starred_arg_seq { ##2 }
481          \seq_gput_right:NV
482              \g__enumext_footnote_starred_int_seq \g__enumext_footnote_starred_int
483          }
484      }
485  \cs_new_protected:Nn \__enumext_print_footnote_mini:
486      {
487      \seq_if_empty:NF \g__enumext_footnote_starred_int_seq
488          {
489          \seq_map_pairwise_function:NNN
490              \g__enumext_footnote_starred_int_seq
491              \g__enumext_footnote_starred_arg_seq
492              \__enumext_footnotetext:nn
493          }
494      \seq_gclear:N \g__enumext_footnote_starred_arg_seq
495      \seq_gclear:N \g__enumext_footnote_starred_int_seq
496      }
```

(*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.

```
497  \cs_new_protected:Nn \__enumext_renew_footnote_standar:
498      {
499      \bool_if:NT \g__enumext_standar_bool
500          {
501          \IfDocumentMetadataTF
502              {
503              \__enumext_renew_footnote:
504              }
505              {
506              \bool_if:NF \l__enumext_footnotes_key_bool
507                  {
508                  \__enumext_renew_footnote:
509                  }
510              }
511          }
512      }
513  \cs_new_protected:Nn \__enumext_print_footnote_standar:
```

```
514     {
515       \bool_if:NT \g__enumext_standar_bool
516         {
517           \IfDocumentMetadataTF
518             {
519               \__enumext_print_footnote:
520             }
521             {
522               \bool_if:NF \l__enumext_footnotes_key_bool
523                 {
524                   \__enumext_print_footnote:
525                 }
526             }
527         }
528     }
```

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.

```
529   \cs_new_protected:Nn \__enumext_renew_footnote_starred:
530     {
531       \IfDocumentMetadataTF
532         {
533           \__enumext_renew_footnote_mini:
534         }
535         {
536           \bool_if:NF \l__enumext_footnotes_key_bool
537             {
538               \__enumext_renew_footnote_mini:
539             }
540         }
541     }
542   \cs_new_protected:Nn \__enumext_print_footnote_starred:
543     {
544       \IfDocumentMetadataTF
545         {
546           \__enumext_print_footnote_mini:
547         }
548         {
549           \bool_if:NF \l__enumext_footnotes_key_bool
550             {
551               \__enumext_print_footnote_mini:
552             }
553         }
554     }
```

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

```
555   \__enumext_after_env:nn { enumext* }
556     {
557       \__enumext_print_footnote_starred:
558     }
559   \__enumext_after_env:nn { keyans* }
560     {
561       \__enumext_print_footnote_starred:
562     }
```

(*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)

```
563   \cs_new_protected:Nn \__enumext_internal_mini_page:
564     {
565       \int_compare:nNnT { \l__enumext_level_int } = { 0 }
```

```
566        {
567          \DeclareDocumentEnvironment{__enumext_mini_page}{ m }
568            {
569               \__enumext_renew_footnote_standar:
570               \__enumext_minipage:w [ t ] { ##1 }
571                 \legacy_if_gset_false:n { @minipage }
572                 \skip_vertical:N \c_zero_skip
573            }
574            {
575                 \skip_vertical:N \c_zero_skip
576               \__enumext_endminipage:
577               \__enumext_print_footnote_standar:
578            }
579          }
580      }
```

(*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.

```
581 \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.

```
582 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
583   {
584     \cs_if_exist:cTF { c@ #2 }
585       { \msg_fatal:nnn { enumext } { counters }{ #2 } }
586       {
587         \tl_set:Nn #1 { #2 }
588         \newcounter { #2 }
589       }
590   }
```

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.

```
591 \__enumext_define_counters:Nn \l__enumext_counter_i_tl    { enumXi     }
592 \__enumext_define_counters:Nn \l__enumext_counter_ii_tl   { enumXii    }
593 \__enumext_define_counters:Nn \l__enumext_counter_iii_tl  { enumXiii   }
594 \__enumext_define_counters:Nn \l__enumext_counter_iv_tl   { enumXiv    }
595 \__enumext_define_counters:Nn \l__enumext_counter_v_tl    { enumXv     }
596 \__enumext_define_counters:Nn \l__enumext_counter_vi_tl   { enumXvi    }
597 \__enumext_define_counters:Nn \l__enumext_counter_vii_tl  { enumXvii   }
598 \__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.

```
599 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
600   {
601     \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
602     \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
603   }
604 \__enumext_register_counter_style:Nn \arabic { 0 }
605 \__enumext_register_counter_style:Nn \Alph   { M }
606 \__enumext_register_counter_style:Nn \alph   { m }
```

```
607 \__enumext_register_counter_style:Nn \Roman  { VIII }
608 \__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.

```
609 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
610   {
611     \hbox_set:Nn \l__enumext_label_width_by_box {#2}
612     \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
613   }
614 \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.

```
615 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
616   {
617     \tl_clear_new:N #1
618     \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
619     \tl_gset_eq:NN \g__enumext_widest_label_tl #1
620     \tl_map_inline:Nn \g__enumext_counter_styles_tl
621       {
622         \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
623         \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
624           { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
625       }
626     \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
627       { \tl_use:N \g__enumext_widest_label_tl }
628     \tl_set_eq:cN { the #2 } #1
629   }
630 \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.

```
631 \cs_set_protected:Npn \__enumext_tmp:n #1
632   {
633     \keys_define:nn { enumext /  #1 }
634       {
635         mode-box .bool_set:N = \l__enumext_mode_box_bool,
636         mode-box .initial:n  = false,
637         mode-box .value_forbidden:n = true,
638       }
639   }
640 \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.

```
641 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
642   {
643     \keys_define:nn { enumext / #1 }
644       {
645         font        .tl_set:c   = { l__enumext_label_font_style_#2_tl },
646         font        .value_required:n = true,
647         labelsep    .dim_set:c  = { l__enumext_labelsep_#2_dim },
648         labelsep    .initial:n  = {0.3333em},
649         labelsep    .value_required:n = true,
```

```
650            labelwidth  .dim_set:c  = { l__enumext_labelwidth_#2_dim },
651            labelwidth  .value_required:n = true,
652            wrap-label  .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
653            wrap-label  .initial:n  = {##1},
654            wrap-label  .value_required:n = true,
655            wrap-label* .code:n = {
656                             \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
657                             \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
658                          },
659            wrap-label* .value_required:n = true,
660          }
661      }
662  \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`.

```
663  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
664    {
665      \keys_define:nn { enumext / #1 }
666        {
667          align .choice:,
668          align / left    .code:n =
669                          {
670                            \tl_clear:c { l__enumext_label_fill_left_#2_tl  }
671                            \tl_set:cn  { l__enumext_label_fill_right_#2_tl } { \hfill }
672                            \str_set:cn { l__enumext_align_label_pos_#2_str } { l }
673                          },
674          align / right   .code:n =
675                          {
676                            \tl_set:cn  { l__enumext_label_fill_left_#2_tl  } { \hfill }
677                            \tl_clear:c { l__enumext_label_fill_right_#2_tl }
678                            \str_set:cn { l__enumext_align_label_pos_#2_str } { r }
679                          },
680          align / center  .code:n =
681                          {
682                            \tl_set:cn { l__enumext_label_fill_left_#2_tl  } { \hfill }
683                            \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
684                            \str_set:cn { l__enumext_align_label_pos_#2_str } { c }
685                          },
686          align / unknown .code:n =
687                          \msg_error:nneee { enumext } { unknown-choice }
688                          { align } { left,~right,~ center } { \exp_not:n {##1} },
689          align .initial:n  = left,
690          align .value_required:n  = true,
691        }
692    }
693  \clist_map_inline:nn
694    {
695      {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
696    }
697    { \__enumext_tmp:nn #1 }

698  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
699    {
700      \keys_define:nn { enumext / #1 }
701        {
702          align .choice:,
703          align / left    .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
704          align / right   .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
705          align / center  .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
706          align / unknown .code:n =
707                          \msg_error:nneee { enumext } { unknown-choice }
708                          { align } { left,~right,~ center } { \exp_not:n {##1} },
709          align .initial:n = left,
710          align .value_required:n  = true,
711        }
712    }
713  \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.

```
714 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
715   {
716     \keys_define:nn { enumext / #1 }
717       {
718         label .code:n    = {
719                             \__enumext_label_style:cvn { l__enumext_label_#2_tl }
720                             { l__enumext_counter_#2_tl } {##1}
721                             \dim_set_eq:cN  { l__enumext_labelwidth_#2_dim }
722                             \l__enumext_current_widest_dim
723                           },
724         label .initial:n = #3,
725         label .value_required:n = true,
726         ref    .code:n   = \__enumext_standar_ref:n {##1},
727         ref    .value_required:n = true,
728       }
729   }
730 \__enumext_tmp:nnn { level-1 } {   i } { \arabic*.}
731 \__enumext_tmp:nnn { level-2 } {  ii } { (\alph*) }
732 \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
733 \__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.

```
734 \cs_new_protected:Npn \__enumext_standar_ref:n #1
735   {
736     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
737     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
738       {
739         \msg_error:nnn { enumext } { key-ref-empty } { enumext }
740       }
741       {
742         \tl_set_eq:Nc
743           \l__enumext_ref_the_count_tl { l__enumext_counter_ \__enumext_level: _tl }
744         \__enumext_regex_counter_style:
745         \tl_set_eq:Nc
746           \l__enumext_ref_the_count_tl { l__enumext_the_counter_ \__enumext_level: _tl }
747         \tl_put_right:ce { l__enumext_renew_the_count_ \__enumext_level: _tl }
748           {
749             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V \l__
750           }
751       }
752   }
```

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

```
753 \cs_new_protected:Nn \__enumext_standar_ref:
754   {
755     \tl_if_empty:cF { l__enumext_renew_the_count_ \__enumext_level: _tl }
756       {
757         \tl_use:c { l__enumext_renew_the_count_ \__enumext_level: _tl }
758       }
759   }
```

(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.

```
760 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
761   {
762     \keys_define:nn { enumext / #1 }
763       {
764         label .code:n    = {
765                             \__enumext_label_style:cvn { l__enumext_label_#2_tl }
766                               { l__enumext_counter_#2_tl } {##1}
767                             \dim_set_eq:cN  { l__enumext_labelwidth_#2_dim }
768                               \l__enumext_current_widest_dim
769                           },
770         label .initial:n = #3,
771         label .value_required:n = true,
772         ref   .code:n    = \__enumext_starred_ref:n {##1},
773         ref   .value_required:n = true,
774       }
775   }
776 \__enumext_tmp:nnn { enumext* } {  vii } { \arabic*.}
777 \__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`.

```
778 \cs_new_protected:Npn \__enumext_starred_ref:n #1
779   {
780     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
781     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
782       {
783         \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
784           {
785             \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
786           }
787           {
788             \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
789             \__enumext_regex_counter_style:
790             \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
791             \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
792               {
793                 \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V
794               }
795           }
796       }
797     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
798       {
799         \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
800           {
801             \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
802           }
803           {
804             \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
805             \__enumext_regex_counter_style:
806             \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
807             \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
808               {
809                 \exp_not:N \renewcommand  { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V
810               }
811           }
812       }
813   }
```

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.

```
814 \cs_new_protected:Nn \__enumext_starred_ref:
815   {
816     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
817       {
818         \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
819           {
820             \tl_use:N \l__enumext_renew_the_count_vii_tl
```

```
821              }
822           }
823        \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
824           {
825              \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
826                 {
827                    \tl_use:N \l__enumext_renew_the_count_viii_tl
828                 }
829           }
830     }
```

(*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.

```
831  \keys_define:nn { enumext / keyans }
832     {
833        label .code:n    = {
834                            \__enumext_label_style:cvn { l__enumext_label_v_tl }
835                              { l__enumext_counter_v_tl } {#1}
836                            \__enumext_label_style:cvn { l__enumext_label_vi_tl }
837                              { l__enumext_counter_vi_tl } {#1}
838                            \dim_set_eq:NN
839                              \l__enumext_labelwidth_v_dim \l__enumext_current_widest_dim
840                            },
841        label .initial:n = \Alph*),
842        label .value_required:n = true,
843        ref   .code:n    = \__enumext_keyans_ref:n {#1},
844        ref   .value_required:n = true,
845     }
```

(*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`.

```
846  \cs_new_protected:Npn \__enumext_keyans_ref:n #1
847     {
848        \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
849        \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
850           {
851              \msg_error:nnn { enumext } { key-ref-empty } { keyans }
852           }
853           {
854              \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
855              \__enumext_regex_counter_style:
856              \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
857              \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
858                 {
859                    \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl } { \exp_not:V \l__
860                 }
861           }
862     }
```

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

```
863  \cs_new_protected:Nn \__enumext_keyans_ref:
864     {
865        \tl_if_empty:NF \l__enumext_renew_the_count_v_tl
866           {
867              \tl_use:N \l__enumext_renew_the_count_v_tl
868           }
869     }
```

(*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.

```
870  \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
871    {
872      \__enumext_if_is_int:nTF { #3 }
873        {
874          \int_set:Nn #2 {#3}
875        }
876        {
877          \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
878            { \int_set:Nn #2 { \int_from_alph:n {#3} } }
879          \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
880            { \int_set:Nn #2  { \int_from_roman:n {#3} } }
881        }
882    }
883  \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.

```
884  \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
885    {
886      \__enumext_if_is_int:nTF {#4}
887        {
888          \setcounter{enumX#1} { #4 }
889        }
890        {
891          \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
892            { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
893          \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
894            { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
895        }
896      \__enumext_label_width_by_box:cv
897        { l__enumext_labelwidth_#1_dim } { l__enumext_label_#1_tl }
898    }
899  \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.

```
900  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
901    {
902      \keys_define:nn { enumext / #1 }
903        {
904          start* .code:n    = {
905                               \__enumext_start_from:ccn
906                                 { l__enumext_label_#2_tl }
907                                 { l__enumext_start_#2_int } {##1}
908                             },
909          start* .value_required:n = true,
910          start  .code:n    = {
911                               \__enumext_start_from:cce
912                                 { l__enumext_label_#2_tl }
913                                 { l__enumext_start_#2_int } { \int_eval:n {##1} }
```

```
914                              },
915          start  .initial:n = 1,
916          start  .value_required:n = true,
917          widest .code:n     = {
918                                  \__enumext_widest_from:nccn {#2}
919                                      { l__enumext_label_#2_tl }
920                                      { l__enumext_labelwidth_#2_dim } {##1}
921                              },
922          widest .value_required:n = true,
923        }
924    }
925  \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.

```
926  \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
927    {
928      \keys_define:nn { enumext / #1 }
929        {
930          topsep    .skip_set:c = { l__enumext_topsep_#2_skip },
931          topsep    .initial:n  = {#3},
932          topsep    .value_required:n = true,
933          partopsep .skip_set:c = l__enumext_partopsep_#2_skip },
934          partopsep .initial:n  = {#4},
935          partopsep .value_required:n = true,
936          parsep    .skip_set:c = { l__enumext_parsep_#2_skip },
937          parsep    .initial:n  = {#5},
938          parsep    .value_required:n = true,
939          itemsep   .skip_set:c = { l__enumext_itemsep_#2_skip },
940          itemsep   .initial:n  = {#6},
941          itemsep   .value_required:n = true,
942          noitemsep .meta:n    = { itemsep = 0pt, parsep = 0pt },
943          noitemsep .value_forbidden:n = true,
944          nosep     .meta:n    = {
945                                  itemsep = 0pt, parsep= 0pt,
946                                  topsep = 0pt, partopsep = 0pt,
947                              },
948          nosep     .value_forbidden:n = true,
949        }
950    }
```

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

```
951  \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
952    { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
953    { 4.0pt plus 2.0pt minus 1.0pt }
954  \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
955    { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
956    { 2.0pt plus 1.0pt minus 1.0pt }
957  \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
958    { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
959  \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
960    { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
961  \__enumext_tmp:nnnnnn { keyans  } { v }{ 4.0pt plus 2.0pt minus 1.0pt }
962    { 2.0pt plus 1.0pt minus 1.0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
963    { 2.0pt plus 1.0pt minus 1.0pt }
964  \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
965    { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
966    { 4.0pt plus 2.0pt minus 1.0pt }
967  \__enumext_tmp:nnnnnn { keyans* } { viii } { 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 }
```

(*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.

```
970  \keys_define:nn { enumext /  level-1 }
971    {
972      base-fix .bool_set:N = \l__enumext_base_line_fix_bool,
973      base-fix .initial:n  = false,
974      base-fix .value_forbidden:n = true,
975    }
```

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.

```
976  \cs_new_protected:Nn \__enumext_nested_base_line_fix:
977    {
978      \bool_lazy_all:nT
979        {
980          { \bool_if_p:N \l__enumext_starred_first_bool }
981          { \bool_if_p:N \l__enumext_base_line_fix_bool }
982          { \bool_not_p:n { \l__enumext_print_keyans_star_bool } }
983        }
984        {
985          \mode_leave_vertical:
986          \vspace { -\dim_eval:n { \baselineskip + \parsep } }
987          \keys_set:nn { enumext / level-1 }
988            {
989              topsep = 0pt, above = 0pt, above* = 0pt,
990            }
991        }
```

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`.

```
992      \bool_lazy_and:nnT
993        { \bool_if_p:N \l__enumext_starred_first_bool }
994        { \bool_if_p:N \l__enumext_print_keyans_star_bool }
995        {
996          \mode_leave_vertical:
997          \skip_vertical:n { -\baselineskip }
998          \skip_vertical:N \c_zero_skip
999          \keys_set:nn { enumext / level-1 }
1000           {
1001             topsep = 0pt, above = 0pt, above* = 0pt,
1002           }
1003       }
1004     \bool_set_false:N \l__enumext_base_line_fix_bool
1005   }
```

(*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.

```
1006  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1007    {
1008      \keys_define:nn { enumext / #1 }
1009        {
1010          itemindent    .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
1011          itemindent    .value_required:n = true,
1012          rightmargin   .dim_set:c = { l__enumext_rightmargin_#2_dim },
1013          rightmargin   .value_required:n = true,
1014          listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
```

```
1015        listparindent .value_required:n = true,
1016        list-offset   .dim_set:c = { l__enumext_listoffset_#2_dim },
1017        list-offset   .value_required:n = true,
1018        list-indent   .code:n    =
1019                       \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
1020                       \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
1021        list-indent   .value_required:n = true,
1022      }
1023    }
1024  \clist_map_inline:nn
1025    {
1026      {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
1027    }
1028    { \__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.

```
1029  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1030    {
1031      \keys_define:nn { enumext / #1 }
1032        {
1033          itemindent    .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
1034          itemindent    .value_required:n = true,
1035          rightmargin   .dim_set:c = { l__enumext_rightmargin_#2_dim },
1036          rightmargin   .value_required:n = true,
1037          listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
1038          listparindent .value_required:n = true,
1039          list-offset   .dim_set:c = { l__enumext_listoffset_#2_dim },
1040          list-offset   .value_required:n = true,
1041          list-indent   .meta:n    = { list-offset = ##1 },
1042          list-indent   .value_required:n = true,
1043        }
1044    }
1045  \clist_map_inline:nn
1046    {
1047      {enumext*}{vii}, {keyans*}{viii}
1048    }
1049    { \__enumext_tmp:nn #1 }
```

13.18.1   Functions for setting the fake itemindent

<div style="float:left">
__enumext_fake_item_indent:
__enumext_keyans_fake_item_indent:
__enumext_fake_item_indent_vii:
__enumext_fake_item_indent_viii:
</div>

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.

```
1050  \cs_set_protected:Nn \__enumext_fake_item_indent:
1051    {
1052      \dim_compare:nNnT
1053        { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
1054        >
1055        { \c_zero_dim }
1056        {
1057          \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
1058            {
1059              \exp_not:N \mode_leave_vertical:
1060              \exp_not:n { \skip_horizontal:n }
1061                { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
1062              \exp_not:N \ignorespaces
1063            }
1064        }
1065    }
1066  \cs_set_protected:Nn \__enumext_keyans_fake_item_indent:
1067    {
1068      \dim_compare:nNnT
1069        { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
1070        {
1071          \tl_set:Ne \l__enumext_fake_item_indent_v_tl
1072            {
1073              \exp_not:N \mode_leave_vertical:
1074              \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
```

```
1075             \exp_not:N \ignorespaces
1076           }
1077       }
1078   }
1079 \cs_set_protected:Nn \__enumext_fake_item_indent_vii:
1080   {
1081     \dim_compare:nNnT
1082       { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
1083       {
1084         \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
1085           {
1086             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
1087             \exp_not:N \ignorespaces
1088           }
1089       }
1090   }
1091 \cs_set_protected:Nn \__enumext_fake_item_indent_viii:
1092   {
1093     \dim_compare:nNnT
1094       { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
1095       {
1096         \tl_set:Ne \l__enumext_fake_item_indent_viii_tl
1097           {
1098             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
1099             \exp_not:N \ignorespaces
1100           }
1101       }
1102   }
```

(*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.

```
1103 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1104   {
1105     \keys_define:nn { enumext / #1 }
1106       {
1107         show-length .bool_set:c = { l__enumext_show_length_#2_bool },
1108         show-length .initial:n  = false,
1109       }
1110   }
1111 \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
before*
after
first

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

```
1112 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1113   {
1114     \keys_define:nn { enumext / #1 }
1115       {
1116         before  .tl_set:c   = { l__enumext_before_no_starred_key_#2_tl },
1117         before  .value_required:n = true,
1118         before* .tl_set:c   = { l__enumext_before_starred_key_#2_tl },
1119         before* .value_required:n = true,
1120         after   .tl_set:c   = { l__enumext_after_stop_list_#2_tl },
1121         after   .value_required:n = true,
1122         first   .tl_set:c   = { l__enumext_after_list_args_#2_tl },
1123         first   .value_required:n = true,
1124       }
1125   }
1126 \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*⟩}.

```
1127 \cs_new_protected:Nn \__enumext_before_args_exec:
1128   {
1129     \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
1130   }
```

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*⟩}}

```
1131 \cs_new_protected:Nn \__enumext_before_keys_exec:
1132   {
1133     \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
1134   }
```

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

```
1135 \cs_new_protected:Nn \__enumext_after_stop_list:
1136   {
1137     \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
1138   }
```

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`.

```
1139 \cs_new_protected:Nn \__enumext_after_args_exec:
1140   {
1141     \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
1142   }
```

(*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.

```
1143 \cs_new_protected:Nn \__enumext_before_args_exec_v:
1144   {
1145     \tl_use:N \l__enumext_before_starred_key_v_tl
1146   }
1147 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
1148   {
1149     \tl_use:N \l__enumext_before_no_starred_key_v_tl
1150   }
1151 \cs_new_protected:Nn \__enumext_after_stop_list_v:
1152   {
1153     \tl_use:N \l__enumext_after_stop_list_v_tl
1154   }
1155 \cs_new_protected:Nn \__enumext_after_args_exec_v:
1156   {
1157     \tl_use:N \l__enumext_after_list_args_v_tl
1158   }
```

(*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.

```
1159 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
1160   {
1161     \tl_use:N \l__enumext_before_starred_key_vii_tl
1162   }
1163 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
1164   {
1165     \tl_use:N \l__enumext_before_starred_key_viii_tl
1166   }
1167 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
1168   {
1169     \tl_use:N \l__enumext_before_no_starred_key_vii_tl
1170   }
1171 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
1172   {
```

```
1173        \tl_use:N \l__enumext_before_no_starred_key_viii_tl
1174      }
1175  \cs_new_protected:Nn \__enumext_after_stop_list_vii:
1176      {
1177        \tl_use:N \l__enumext_after_stop_list_vii_tl
1178      }
1179  \cs_new_protected:Nn \__enumext_after_stop_list_viii:
1180      {
1181        \tl_use:N \l__enumext_after_stop_list_viii_tl
1182      }
1183  \cs_new_protected:Nn \__enumext_after_args_exec_vii:
1184      {
1185        \tl_use:N \l__enumext_after_list_args_vii_tl
1186      }
1187  \cs_new_protected:Nn \__enumext_after_args_exec_viii:
1188      {
1189        \tl_use:N \l__enumext_after_list_args_viii_tl
1190      }
```

(*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.

```
1191  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1192      {
1193        \keys_define:nn { enumext / #1 }
1194          {
1195            mini-env     .dim_set:c  = { l__enumext_minipage_right_#2_dim },
1196            mini-env     .value_required:n = true,
1197            mini-sep     .dim_set:c  = { l__enumext_minipage_hsep_#2_dim },
1198            mini-sep     .initial:n  = 0.3333em,
1199            mini-sep     .value_required:n = true,
1200            columns-sep .dim_set:c  = { l__enumext_columns_sep_#2_dim },
1201            columns-sep .value_required:n = true,
1202            columns      .int_set:c  = { l__enumext_columns_#2_int },
1203            columns      .initial:n  = 1,
1204            columns      .value_required:n = true,
1205          }
1206      }
1207  \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.

```
1208  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1209      {
1210        \keys_define:nn { enumext / #1 }
1211          {
1212            mini-right  .tl_gset:c = { g__enumext_miniright_code_#2_tl },
1213            mini-right  .value_required:n = true,
1214            mini-right* .code:n    = {
1215                                      \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
1216                                      \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1217                                    },
1218            mini-right* .value_required:n = true,
1219          }
1220      }
1221  \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*.

```
1222 \cs_new_protected:Nn \__enumext_multi_set_vskip:
1223   {
1224     \skip_set:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1225       {
1226         \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1227       }
1228     \skip_set:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1229       {
1230         \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1231       }
1232     \__enumext_add_pre_parsep:
1233   }
```

(*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*.

```
1234 \cs_new_protected:Nn \__enumext_add_pre_parsep:
1235   {
1236     \int_case:nn { \l__enumext_level_int }
1237       {
1238         { 2 }{
1239             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1240               {
1241                 \skip_add:Nn \l__enumext_multicols_above_ii_skip
1242                   {
1243                     \l__enumext_parsep_i_skip
1244                   }
1245               }
1246           }
1247         { 3 }{
1248             \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1249               {
1250                 \skip_add:Nn \l__enumext_multicols_above_iii_skip
1251                   {
1252                     \l__enumext_parsep_ii_skip
1253                   }
1254               }
1255           }
1256         { 4 }{
1257             \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1258               {
1259                 \skip_add:Nn \l__enumext_multicols_above_iv_skip
1260                   {
1261                     \l__enumext_parsep_iii_skip
1262                   }
```

```
1263                          }
1264                       }
1265                    }
1266                 }
```

(*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*⟩.

```
1267  \cs_new_protected:Nn \__enumext_multi_addvspace:
1268    {
1269      \__enumext_multi_set_vskip:
1270      \mode_if_vertical:T
1271        {
1272          \skip_add:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1273            {
1274              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1275            }
1276          \skip_add:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1277            {
1278              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1279            }
1280        }
1281      \par\nopagebreak
1282      \addvspace{ \skip_use:c { l__enumext_multicols_above_ \__enumext_level: _skip } }
1283    }
```

(*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.

```
1284  \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1285    {
1286      \skip_set:Nn \l__enumext_multicols_above_v_skip
1287        {
1288          \l__enumext_topsep_v_skip
1289        }
1290      \skip_set:Nn \l__enumext_multicols_below_v_skip
1291        {
1292          \l__enumext_topsep_v_skip
1293        }
1294    }
1295  \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1296    {
1297      \__enumext_keyans_multi_set_vskip:
1298      \mode_if_vertical:T
1299        {
1300          \skip_add:Nn \l__enumext_multicols_above_v_skip
1301            {
1302              \skip_use:N \l__enumext_partopsep_v_skip
1303            }
1304          \skip_add:Nn \l__enumext_multicols_below_v_skip
1305            {
1306              \skip_use:N \l__enumext_partopsep_v_skip
1307            }
1308        }
1309      \par\nopagebreak
1310      \addvspace{ \l__enumext_multicols_above_v_skip }
1311    }
```

(*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`.

```
1312 \cs_new_protected:Nn \__enumext_minipage_set_skip:
1313   {
1314     \skip_set:Nn \l__enumext_minipage_right_skip
1315       {
1316         \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1317       }
1318     \mode_if_vertical:T
1319       {
1320         \skip_add:Nn \l__enumext_minipage_right_skip
1321           {
1322             \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1323           }
1324       }
1325     \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:`.

```
1326     \skip_set_eq:cN
1327       { l__enumext_multicols_above_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1328     \skip_set_eq:cN
1329       { l__enumext_multicols_below_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1330     \__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`.

```
1331     \int_compare:nNnT
1332       { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1333       {
1334         \skip_zero:N \topskip
1335         \skip_set_eq:Nc \multicolsep { l__enumext_multicols_above_ \__enumext_level: _skip }
1336       }
1337   }
```

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*.

```
1338 \cs_new_protected:Nn \__enumext_minipage_add_space:
1339   {
1340     \__enumext_minipage_set_skip:
1341     \__enumext_unskip_unkern:
1342     \mode_if_vertical:TF
1343       {
1344         \nopagebreak\nointerlineskip
```

```
1345           }
1346         {
1347           \par\nopagebreak\nointerlineskip
1348           \skip_zero:c { l__enumext_partopsep_ \__enumext_level: _skip }
1349         }
1350       \int_compare:nNnTF
1351         { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1352         {
1353           \addvspace{ 0.445\box_ht:N \strutbox }
1354         }
1355         {
1356           \addvspace{ 0.250\box_ht:N \strutbox }
1357         }
1358     }
```

(*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.

```
1359  \cs_new_protected:Nn \__enumext_pre_itemsep_skip:
1360    {
1361      \int_case:nn { \l__enumext_level_int }
1362        {
1363          { 2 }{
1364              \skip_if_eq:nnTF
1365                { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1366                {
1367                  \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1368                  \skip_set:Nn \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1369                }
1370                {
1371                  \dim_compare:nNnT
1372                    { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1373                    {
1374                      \skip_sub:Nn
1375                        \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1376                      \skip_sub:Nn
1377                        \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1378                      \skip_add:Nn
1379                        \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1380                      \skip_add:Nn
1381                        \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1382                    }
1383                  \dim_compare:nNnT
1384                    { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1385                    {
1386                      \skip_set:Nn \l__enumext_minipage_temp_skip
1387                        {
1388                          \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1389                        }
1390                      \skip_sub:Nn
1391                        \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1392                      \skip_sub:Nn
1393                        \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1394                      \skip_add:Nn
1395                        \l__enumext_minipage_after_skip
1396                        { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1397                      \skip_add:Nn
1398                        \l__enumext_multicols_below_ii_skip
1399                        { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1400                    }
1401                }
1402            }
1403          { 3 }{
1404              \skip_if_eq:nnTF
1405                { \l__enumext_itemsep_ii_skip } { \c_zero_skip }
1406                {
1407                  \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1408                  \skip_set:Nn \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1409                }
```

```
1410                              {
1411                                  \dim_compare:nNnT
1412                                      { \l__enumext_itemsep_ii_skip } < { \l__enumext_minipage_after_skip }
1413                                      {
1414                                          \skip_sub:Nn
1415                                              \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1416                                          \skip_sub:Nn
1417                                              \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1418                                          \skip_add:Nn
1419                                              \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1420                                          \skip_add:Nn
1421                                              \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1422                                      }
1423                                  \dim_compare:nNnT
1424                                      { \l__enumext_itemsep_ii_skip } > { \l__enumext_minipage_after_skip }
1425                                      {
1426                                          \skip_set:Nn \l__enumext_minipage_temp_skip
1427                                              {
1428                                                  \l__enumext_itemsep_ii_skip - \l__enumext_minipage_after_skip
1429                                              }
1430                                          \skip_sub:Nn
1431                                              \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1432                                          \skip_sub:Nn
1433                                              \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1434                                          \skip_add:Nn
1435                                              \l__enumext_minipage_after_skip
1436                                              { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1437                                          \skip_add:Nn
1438                                              \l__enumext_multicols_below_iii_skip
1439                                              { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1440                                      }
1441                              }
1442                          }
1443              { 4 }{
1444                      \skip_if_eq:nnTF { \l__enumext_itemsep_iii_skip } { \c_zero_skip }
1445                          {
1446                              \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1447                              \skip_set:Nn \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1448                          }
1449                          {
1450                              \dim_compare:nNnT
1451                                  { \l__enumext_itemsep_iii_skip } < { \l__enumext_minipage_after_skip }
1452                                  {
1453                                      \skip_sub:Nn
1454                                          \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1455                                      \skip_sub:Nn
1456                                          \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1457                                      \skip_add:Nn
1458                                          \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1459                                      \skip_add:Nn
1460                                          \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1461                                  }
1462                              \dim_compare:nNnT
1463                                  { \l__enumext_itemsep_iii_skip } > { \l__enumext_minipage_after_skip }
1464                                  {
1465                                      \skip_set:Nn \l__enumext_minipage_temp_skip
1466                                          {
1467                                              \l__enumext_itemsep_iii_skip - \l__enumext_minipage_after_skip
1468                                          }
1469                                      \skip_sub:Nn
1470                                          \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1471                                      \skip_sub:Nn
1472                                          \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1473                                      \skip_add:Nn
1474                                          \l__enumext_minipage_after_skip
1475                                          { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1476                                      \skip_add:Nn
1477                                          \l__enumext_multicols_below_iv_skip
1478                                          { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1479                                  }
1480                          }
```

```
1481                    }
1482                }
1483        }
```

(*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.

```
1484  \cs_new_protected:Nn \__enumext_keyans_minipage_set_skip:
1485    {
1486      \skip_zero:N \l__enumext_minipage_after_skip
1487      \skip_zero:N \l__enumext_minipage_left_skip
1488      \skip_zero:N \l__enumext_minipage_right_skip
1489      \skip_set:Nn \l__enumext_minipage_right_skip
1490        {
1491          \l__enumext_topsep_v_skip
1492        }
1493      \mode_if_vertical:T
1494        {
1495          \skip_add:Nn \l__enumext_minipage_right_skip
1496            {
1497              \l__enumext_partopsep_v_skip
1498            }
1499        }
1500      \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip
1501      \skip_set_eq:NN \l__enumext_multicols_above_v_skip \l__enumext_minipage_right_skip
1502      \skip_set_eq:NN \l__enumext_multicols_below_v_skip \l__enumext_minipage_right_skip
1503      \__enumext_keyans_pre_itemsep_skip:
1504      \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
1505        {
1506          \skip_zero:N \topskip
1507          \skip_set_eq:NN \multicolsep \l__enumext_minipage_right_skip
1508        }
1509    }
1510  \cs_new_protected:Nn \__enumext_keyans_minipage_add_space:
1511    {
1512      \__enumext_keyans_minipage_set_skip:
1513      \__enumext_unskip_unkern:
1514      \mode_if_vertical:TF
1515        {
1516          \nopagebreak\nointerlineskip
1517        }
1518        {
1519          \par\nopagebreak\nointerlineskip
1520          \skip_zero:N \l__enumext_partopsep_v_skip
1521        }
1522      \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1523        {
1524          \addvspace{ 0.445\box_ht:N \strutbox }
1525        }
1526        {
1527          \addvspace{ 0.250\box_ht:N \strutbox }
1528        }
1529    }
1530  \cs_new_protected:Nn \__enumext_keyans_pre_itemsep_skip:
1531    {
1532      \skip_if_eq:nnTF
1533        { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1534        {
1535          \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1536          \skip_set:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1537        }
1538        {
1539          \dim_compare:nNnT
1540            { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1541            {
1542              \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1543              \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1544              \skip_add:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
```

```
1545              \skip_add:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1546            }
1547          \dim_compare:nNnT
1548            { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1549            {
1550              \skip_set:Nn \l__enumext_minipage_temp_skip
1551                {
1552                  \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1553                }
1554              \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1555              \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1556              \skip_add:Nn \l__enumext_minipage_after_skip
1557                { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1558              \skip_add:Nn \l__enumext_multicols_below_v_skip
1559                { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1560            }
1561        }
1562    }
```

(*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:
__enumext_mini_set_vskip_viii:

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

```
1563  \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1564    {
1565      \skip_zero_new:N \l__enumext_minipage_left_skip
1566      \skip_gzero_new:N \g__enumext_minipage_right_skip
1567      \skip_gzero_new:N \g__enumext_minipage_after_skip
1568      \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1569        {
1570          \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1571          \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1572        }
1573        {
1574          \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1575          \skip_gset:Nn \g__enumext_minipage_right_skip
1576            {
1577              \l__enumext_topsep_vii_skip
1578            }
1579          \skip_gset:Nn \g__enumext_minipage_after_skip
1580            {
1581              0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1582            }
1583        }
1584    }
1585  \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1586    {
1587      \skip_zero_new:N \l__enumext_minipage_after_skip
1588      \skip_zero_new:N \l__enumext_minipage_left_skip
1589      \skip_zero_new:N \l__enumext_minipage_right_skip
1590      \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1591        {
1592          \skip_set:Nn \l__enumext_minipage_left_skip
1593            {
1594              0.5\box_dp:N \strutbox
1595            }
1596          \skip_set:Nn \l__enumext_minipage_right_skip
1597            {
1598              \l__enumext_partopsep_viii_skip
1599            }
1600          \skip_set:Nn \l__enumext_minipage_after_skip
1601            {
1602              1.6\box_dp:N \strutbox
1603            }
1604        }
1605        {
1606          \skip_set:Nn \l__enumext_minipage_left_skip
1607            {
```

```
1608              0.5875\box_dp:N \strutbox
1609          }
1610      \skip_set:Nn \l__enumext_minipage_right_skip
1611        {
1612          \l__enumext_topsep_viii_skip
1613        }
1614      \skip_set:Nn \l__enumext_minipage_after_skip
1615        {
1616          0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1617        }
1618    }
1619  }
```

(*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.

```
1620  \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1621    {
1622      \__enumext_mini_set_vskip_vii:
1623      \par\nopagebreak
1624      \addvspace { \l__enumext_minipage_left_skip }
1625    }
1626  \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1627    {
1628      \__enumext_mini_set_vskip_viii:
1629      \par\nopagebreak
1630      \addvspace { \l__enumext_minipage_left_skip }
1631    }
```

(*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.

```
1632  \NewDocumentCommand \miniright { s }
1633    {
1634      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1635        {
1636          \msg_error:nnn { enumext } { wrong-miniright-place }
1637        }
1638      % outside
1639      \bool_lazy_and:nnT
1640        { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
1641        { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
1642        {
1643          \msg_error:nnn { enumext } { wrong-miniright-place }
1644        }
1645      % starred env
1646      \bool_lazy_and:nnT
1647        { \bool_if_p:N \g__enumext_starred_bool }
1648        { \bool_not_p:n { \l__enumext_standar_bool } }
1649        {
1650          \msg_error:nnn { enumext } { wrong-miniright-starred }
1651        }
1652      % exec
1653      \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
1654        {
1655          \__enumext_keyans_mini_right_cmd:n {#1}
1656        }
```

```
1657          { \__enumext_mini_right_cmd:n {#1} }
1658        }
```

(*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.

```
1659  \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1660    {
1661      \dim_compare:nNnTF
1662        { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1663        {
1664          \__enumext_multicols_stop:
1665          \int_compare:nNnT
1666            { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } = { 1 }
1667            {
1668              \par\addvspace{ \l__enumext_minipage_after_skip }
1669            }
1670          \end__enumext_mini_page
1671          \hfill
1672          \__enumext_mini_page{ \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } }
1673            \par\nointerlineskip
1674            \addvspace { \l__enumext_minipage_right_skip }
1675            \bool_if:nF {#1}
1676              {
1677                \centering
1678              }
1679            \int_gzero:N \g__enumext_minipage_stat_int
1680        }
1681        { \msg_error:nnn { enumext } { wrong-miniright-use } }
1682      % paranoia
1683      \RenewDocumentCommand \miniright { s }
1684        {
1685          \msg_error:nn { enumext } { many-miniright-used }
1686        }
1687    }
```

(*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.

```
1688  \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1689    {
1690      \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1691        {
1692          \__enumext_keyans_multicols_stop:
1693          \int_compare:nNnT { \l__enumext_columns_v_int } = { 1 }
1694            {
1695              \par\addvspace{ \l__enumext_minipage_after_skip }
1696            }
1697          \end__enumext_mini_page
1698          \hfill
1699          \__enumext_mini_page{ \l__enumext_minipage_right_v_dim }
1700            \par\nointerlineskip
1701            \addvspace { \l__enumext_minipage_right_skip }
1702            \bool_if:nF {#1}
1703              {
1704                \centering
1705              }
1706            \int_gzero:N \g__enumext_minipage_stat_int
1707        }
1708        { \msg_error:nnn { enumext } { wrong-miniright-use } }
1709      % paranoia
1710      \RenewDocumentCommand \miniright { s }
1711        {
1712          \msg_error:nn { enumext } { many-miniright-used }
```

```
1713        }
1714    }
```

(*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.

```
1715 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1716   {
1717     \keys_define:nn { enumext / #1 }
1718       {
1719         above  .skip_set:c = { l__enumext_vspace_above_#2_skip },
1720         above  .value_required:n = true,
1721         above* .code:n     = \bool_set_true:c { l__enumext_vspace_a_star_#2_bool }
1722                              \keys_set:nn { enumext / #1 } { above = {##1} },
1723         above* .value_required:n = true,
1724         below  .skip_set:c = { l__enumext_vspace_below_#2_skip },
1725         below  .value_required:n = true,
1726         below* .code:n     = \bool_set_true:c { l__enumext_vspace_b_star_#2_bool }
1727                              \keys_set:nn { enumext / #1 } { below = {##1} },
1728         below* .value_required:n = true,
1729       }
1730   }
1731 \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.

```
1732 \cs_new_protected:Nn \__enumext_vspace_above:
1733   {
1734     \skip_if_eq:nnF
1735       { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1736       {
1737         \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
1738           {
1739             \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1740           }
1741           {
1742             \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1743           }
1744       }
1745   }
```

(*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.

```
1746 \cs_new_protected:Nn \__enumext_vspace_below:
1747   {
1748     \skip_if_eq:nnF
1749       { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1750       {
1751         \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1752           {
1753             \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1754           }
1755           {
1756             \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1757           }
1758       }
1759   }
```

(*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.

```
1760 \cs_new_protected:Nn \__enumext_vspace_above_v:
1761   {
1762     \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1763       {
1764         \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1765           {
1766             \vspace*{ \l__enumext_vspace_above_v_skip }
1767           }
1768           { \vspace { \l__enumext_vspace_above_v_skip } }
1769       }
1770   }
```

(*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.

```
1771 \cs_new_protected:Nn \__enumext_vspace_below_v:
1772   {
1773     \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1774       {
1775         \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1776           {
1777             \vspace*{ \l__enumext_vspace_below_v_skip }
1778           }
1779           { \vspace { \l__enumext_vspace_below_v_skip } }
1780       }
1781   }
```

(*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.

```
1782 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1783   {
1784     \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1785       {
1786         \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1787           {
1788             \vspace*{ \l__enumext_vspace_above_vii_skip }
1789           }
1790           { \vspace { \l__enumext_vspace_above_vii_skip } }
1791       }
1792   }
1793 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1794   {
1795     \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1796       {
1797         \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1798           {
1799             \vspace*{ \l__enumext_vspace_above_viii_skip }
1800           }
1801           { \vspace { \l__enumext_vspace_above_viii_skip } }
1802       }
1803   }
```

(*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.

```
1804 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1805   {
1806     \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1807       {
1808         \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1809           {
1810             \vspace*{ \l__enumext_vspace_below_vii_skip }
```

```
1811              }
1812            { \vspace { \l__enumext_vspace_below_vii_skip } }
1813        }
1814    }
1815 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1816    {
1817      \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1818        {
1819          \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1820            {
1821              \vspace*{ \l__enumext_vspace_below_viii_skip }
1822            }
1823            { \vspace { \l__enumext_vspace_below_viii_skip } }
1824        }
1825    }
```

(*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
resume
resume*

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

```
1826 \cs_set_protected:Npn \__enumext_tmp:n #1
1827    {
1828      \keys_define:nn { enumext / #1 }
1829        {
1830          series  .str_set:N = \l__enumext_series_str,
1831          series  .value_required:n = true,
1832          resume  .code:n = \__enumext_resume_series:n {##1},
1833          resume* .code:n = \__enumext_resume_starred:,
1834          resume* .value_forbidden:n = true,
1835        }
1836    }
1837 \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
__enumext_filter_series_key:n
__enumext_filter_series_pair:nn

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

```
1838 \cs_new:Npn \__enumext_filter_series:n #1
1839    {
1840      \use:e
1841        {
1842          \keyval_parse:NNn
1843            \__enumext_filter_series_key:n
1844            \__enumext_filter_series_pair:nn {#1}
1845        }
1846    }
```

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.

```
1847 \cs_new:Npn \__enumext_filter_series_key:n #1
1848    {
1849      \str_case:nnF {#1}
1850        {
1851          { resume } {} { resume* } {} { base-fix } {}
1852        }
1853        { , { \exp_not:n {#1} } }
1854    }
```

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.

```
1855 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1856    {
1857      \str_case:nnF {#1}
1858        {
1859          { series } {} { resume } {} { start } {}
```

```
1860            { start* } {}  { save-ans } {} { save-key } {}
1861          }
1862          { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1863    }
```

(*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).

```
1864 \cs_new_protected:Npn \__enumext_parse_series:n #1
1865   {
1866     \str_if_empty:NTF \l__enumext_series_str
1867       {
1868         \bool_if:NF \l__enumext_resume_active_bool
1869           {
1870             \__enumext_resume_last:n {#1}
1871           }
1872       }
1873       {
1874         \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str _tl }
1875         \tl_gset:ce { g__enumext_series_ \l__enumext_series_str _tl }
1876           { \__enumext_filter_series:n {#1} }
1877         \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str _int }
1878           {
1879             \int_new:c { g__enumext_series_ \l__enumext_series_str _int }
1880           }
1881       }
1882   }
```

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.

```
1883 \cs_new_protected:Npn \__enumext_resume_last:n #1
1884   {
1885     \bool_if:NT \l__enumext_standar_first_bool
1886       {
1887         \tl_gclear:N \g__enumext_standar_series_tl
1888         \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1889       }
1890     \bool_if:NT \l__enumext_starred_first_bool
1891       {
1892         \tl_gclear:N \g__enumext_starred_series_tl
1893         \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1894       }
1895   }
```

(*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).

```
1896 \cs_new_protected:Nn \__enumext_resume_save_counter:
1897   {
```

```
1898    \bool_if:NT \g__enumext_standar_bool
1899      {
1900        \tl_if_empty:NF \l__enumext_series_str
1901          {
1902            \int_gset_eq:cN
1903              { g__enumext_series_ \l__enumext_series_str _int } \value{enumXi}
1904          }
1905        \tl_if_empty:NTF \l__enumext_resume_name_tl
1906          {
1907            \str_if_empty:NT \l__enumext_series_str
1908              {
1909                \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1910              }
1911          }
1912          {
1913            \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1914              {
1915                \int_gset_eq:cN
1916                  { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXi}
1917              }
1918          }
1919        \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1920          {
1921            \int_gset_eq:cN
1922              { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1923          }
1924      }
1925    \bool_if:NT \g__enumext_starred_bool
1926      {
1927        \tl_if_empty:NF \l__enumext_series_str
1928          {
1929            \int_gset_eq:cN
1930              { g__enumext_series_ \l__enumext_series_str _int } \value{enumXvii}
1931          }
1932        \tl_if_empty:NTF \l__enumext_resume_name_tl
1933          {
1934            \str_if_empty:NT \l__enumext_series_str
1935              {
1936                \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1937              }
1938          }
1939          {
1940            \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1941              {
1942                \int_gset_eq:cN
1943                  { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXvii}
1944              }
1945          }
1946        \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1947          {
1948            \int_gset_eq:cN
1949              { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1950          }
1951      }
1952  }
```

(*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.

```
1953  \cs_new_protected:Npn \__enumext_resume_series:n #1
1954    {
1955      \tl_if_empty:nTF {#1}
1956        {
1957          \__enumext_resume_counter:n { }
```

```
1958              }
1959              {
1960                \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1961                  {
1962                    \__enumext_resume_counter:n {#1}
1963                    \bool_if:NT \g__enumext_standar_bool
1964                      {
1965                        \keys_set:nv { enumext / level-1 }
1966                          { g__enumext_series_ \tl_to_str:n {#1} _tl }
1967                      }
1968                    \bool_if:NT \g__enumext_starred_bool
1969                      {
1970                        \keys_set:nv { enumext / enumext* }
1971                          { g__enumext_series_ \tl_to_str:n {#1} _tl }
1972                      }
1973                  }
1974                  {
1975                    \bool_if:NT \g__enumext_standar_bool
1976                      {
1977                        \msg_error:nnn { enumext } { unknown-series } {#1}
1978                      }
1979                    \bool_if:NT \g__enumext_starred_bool
1980                      {
1981                        \msg_error:nnn { enumext } { unknown-series } {#1}
1982                      }
1983                  }
1984              }
1985          }
```

(*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.

```
1986 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1987    {
1988      \bool_set_true:N \l__enumext_resume_active_bool
1989      \tl_set:Nn \l__enumext_resume_name_tl {#1}
1990      \tl_if_empty:NTF \l__enumext_resume_name_tl
1991        {
1992          \__enumext_resume_counter:
1993        }
1994        {
1995          \__enumext_resume_counter_series:
1996        }
1997      \__enumext_resume_counter_save_ans:
1998    }
```

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.

```
1999 \cs_new_protected:Nn \__enumext_resume_counter:
2000    {
2001      \bool_if:NT \g__enumext_standar_bool
2002        {
2003          \int_gincr:N \g__enumext_resume_int
2004          \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
2005        }
2006      \bool_if:NT \g__enumext_starred_bool
2007        {
2008          \int_gincr:N \g__enumext_resume_vii_int
2009          \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
2010        }
2011    }
```

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.

```
2012  \cs_new_protected:Nn \__enumext_resume_counter_series:
2013    {
2014      \bool_if:NT \g__enumext_standar_bool
2015        {
2016          \int_set:Nn \l__enumext_start_i_int
2017            {
2018              \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
2019            }
2020        }
2021      \bool_if:NT \g__enumext_starred_bool
2022        {
2023          \int_set:Nn \l__enumext_start_vii_int
2024            {
2025              \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
2026            }
2027        }
2028    }
```

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.

```
2029  \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
2030    {
2031      \bool_lazy_and:nnT
2032        { \bool_if_p:N \l__enumext_standar_first_bool }
2033        { \bool_if_p:N \l__enumext_store_active_bool }
2034        {
2035          \int_set:Nn \l__enumext_start_i_int
2036            {
2037              \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
2038            }
2039        }
2040      \bool_lazy_and:nnT
2041        { \bool_if_p:N \l__enumext_starred_first_bool }
2042        { \bool_if_p:N \l__enumext_store_active_bool }
2043        {
2044          \int_set:Nn \l__enumext_start_vii_int
2045            {
2046              \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
2047            }
2048        }
2049    }
```

(*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.

```
2050  \cs_new_protected:Nn \__enumext_resume_starred:
2051    {
2052      \bool_if:NT \g__enumext_standar_bool
2053        {
2054          \tl_if_empty:NF \g__enumext_standar_series_tl
2055            {
2056              \__enumext_resume_counter:n { }
2057              \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
2058            }
2059        }
2060      \bool_if:NT \g__enumext_starred_bool
2061        {
2062          \tl_if_empty:NF \g__enumext_starred_series_tl
2063            {
2064              \__enumext_resume_counter:n { }
2065              \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
2066            }
2067        }
2068    }
```

(*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*.

```
2069 \cs_set_protected:Npn \__enumext_tmp:n #1
2070   {
2071     \keys_define:nn { enumext / #1 }
2072       {
2073         save-ans .code:n = \__enumext_storing_set:n {##1},
2074         save-ans .value_required:n = true,
2075       }
2076   }
2077 \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: The functions __enumext_start_save_ans_msg: and __enumext_stop_save_ans_msg: will display
__enumext_stop_save_ans_msg: in the terminal and .log file the environment in which the save-ans key was executed along with the line at the beginning and end of it. The function __enumext_start_save_ans_msg: will be passed to __enumext_storing_set:n and the function __enumext_stop_save_ans_msg: will be passed to the function __enumext_execute_after_env:.

```
2078 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
2079   {
2080     \msg_term:nnVV { enumext } { save-ans-log }
2081       \g__enumext_envir_name_tl \l__enumext_store_name_tl
2082   }
2083 \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
2084   {
2085     \msg_term:nnVV { enumext } { save-ans-log-hook }
2086       \g__enumext_envir_name_tl \g__enumext_store_name_tl
2087   }
```

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

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

```
2088 \cs_new_protected:Npn \__enumext_storing_set:n #1
2089   {
2090     \tl_set:Ne \l__enumext_store_name_tl {#1}
2091     \tl_if_empty:NTF \l__enumext_store_name_tl
2092       {
2093         \bool_lazy_or:nnT
2094           { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
2095           {
2096             \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
2097           }
2098       }
2099       {
2100         \bool_lazy_or:nnT
2101           { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
2102           {
2103             \__enumext_start_save_ans_msg:
2104             \__enumext_storing_exec:
2105           }
2106       }
2107   }
```

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.

```
2108 \cs_new_protected:Nn \__enumext_storing_exec:
2109   {
```

```
2110        \bool_set_true:N \l__enumext_store_active_bool
2111        \bool_set_true:N \l__enumext_check_answers_bool
2112        \tl_gset:NV \g__enumext_store_name_tl \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*`.

```
2113        \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
2114          {
2115            \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
2116            \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
2117          }
2118        \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
2119          {
2120            \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
2121            \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
2122          }
2123        \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
2124          {
2125            \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
2126            \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
2127          }
2128      }
```

(*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.

```
2129 \cs_set_protected:Npn \__enumext_tmp:n #1
2130   {
2131      \keys_define:nn { enumext / #1 }
2132        {
2133           check-ans .bool_set:N = \l__enumext_check_ans_key_bool,
2134           check-ans .initial:n  = false,
2135           check-ans .value_required:n = true,
2136           no-store   .code:n = {
2137                            \bool_set_false:N \l__enumext_check_answers_bool
2138                            \bool_set_false:N \l__enumext_check_ans_key_bool
2139                          },
```

```
2140          no-store  .value_forbidden:n = true,
2141        }
2142    }
2143  \clist_map_inline:nn
2144    {
2145      level-1, level-2, level-3, level-4, enumext*
2146    }
2147    { \__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.

```
2148  \cs_new_protected:Nn \__enumext_check_ans_active:
2149    {
2150      \tl_if_empty:NF \l__enumext_store_name_tl
2151        {
2152          \bool_if:NT \l__enumext_check_answers_bool
2153            {
2154              \__enumext_check_ans_level:
2155            }
2156        }
2157    }
```

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*".

```
2158  \cs_new_protected:Nn \__enumext_check_ans_level:
2159    {
2160      \int_case:nn { \l__enumext_level_int }
2161        {
2162          { 1 }{
2163              \bool_lazy_all:nT
2164                {
2165                  { \bool_if_p:N \g__enumext_starred_bool }
2166                  { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
2167                }
2168                {
2169                  \int_gdecr:N \g__enumext_item_number_int
2170                  \bool_set_false:N \l__enumext_item_number_bool
2171                }
2172            }
2173          { 2 }{
2174              \int_gdecr:N \g__enumext_item_number_int
2175              \bool_set_false:N \l__enumext_item_number_bool
2176            }
2177          { 3 }{
2178              \int_gdecr:N \g__enumext_item_number_int
2179              \bool_set_false:N \l__enumext_item_number_bool
2180            }
2181          { 4 }{
2182              \int_gdecr:N \g__enumext_item_number_int
2183              \bool_set_false:N \l__enumext_item_number_bool
2184            }
2185        }
```

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.

```
2186      \int_case:nn { \l__enumext_level_h_int }
2187        {
2188          { 1 }{
2189              \bool_lazy_all:nT
2190                {
2191                  { \bool_if_p:N \g__enumext_standar_bool }
2192                  { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
2193                }
2194                {
```

```
2195                        \int_gdecr:N \g__enumext_item_number_int
2196                        \bool_set_false:N \l__enumext_item_number_bool
2197                      }
2198                  }
2199              }
2200          }
```

(*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.

```
2201  \cs_new_protected:Nn \__enumext_check_ans_key_hook:
2202    {
2203      \bool_lazy_and:nnT
2204        { \bool_if_p:N \l__enumext_check_ans_key_bool }
2205        { \bool_if_p:N \g__enumext_standar_bool }
2206        {
2207          \bool_gset_true:N \g__enumext_check_ans_key_bool
2208        }
2209      \bool_lazy_and:nnT
2210        { \bool_if_p:N \l__enumext_check_ans_key_bool }
2211        { \bool_if_p:N \g__enumext_starred_bool }
2212        {
2213          \bool_gset_true:N \g__enumext_check_ans_key_bool
2214        }
2215    }
```

(*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:.

```
2216  \cs_new_protected:Nn \__enumext_item_answer_diff:
2217    {
2218      \int_gset:Nn \g__enumext_item_answer_diff_int
2219        {
2220          \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
2221        }
2222    }
```

(*End of definition for* __enumext_item_answer_diff:.)

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

```
2223  \cs_new_protected:Nn \__enumext_check_ans_show:
2224    {
2225      \int_case:nn { \g__enumext_item_answer_diff_int }
2226        {
2227          { -1 }{ \__enumext_check_ans_msg_less:      }
2228          {  0 }{ \__enumext_check_ans_msg_same_ok: }
2229          {  1 }{ \__enumext_check_ans_msg_greater: }
2230        }
2231    }
2232  \cs_new_protected:Nn \__enumext_check_ans_msg_less:
2233    {
2234      \msg_warning:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2235        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2236    }
2237  \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
2238    {
2239      \msg_term:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2240        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2241    }
2242  \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
2243    {
2244      \msg_warning:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2245        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2246    }
```

(*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:`.

```
2247  \cs_new_protected:Nn \__enumext_check_ans_log:
2248    {
2249      \int_case:nn { \g__enumext_item_answer_diff_int }
2250        {
2251          { -1 }{ \__enumext_check_ans_log_msg_less:    }
2252          {  0 }{ \__enumext_check_ans_log_msg_same_ok: }
2253          {  1 }{ \__enumext_check_ans_log_msg_greater: }
2254        }
2255    }
2256  \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
2257    {
2258      \msg_log:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2259        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2260    }
2261  \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
2262    {
2263      \msg_log:nneee { enumext } { items-same-answer }  { \g__enumext_store_name_tl }
2264        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2265    }
2266  \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
2267    {
2268      \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2269        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2270    }
```

(*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.

```
2271  \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
2272    {
2273      \int_compare:nNnT
2274        { \g__enumext_check_starred_cmd_int } = { 0 }
2275        {
2276          \msg_warning:nnnV
2277            { enumext } { missing-starred }{ #1 } \l__enumext_check_start_line_env_tl
2278        }
2279      \int_compare:nNnT
2280        { \g__enumext_check_starred_cmd_int } > { 1 }
2281        {
2282          \msg_warning:nnnV
2283            { enumext } { many-starred }{ #1 } \l__enumext_check_start_line_env_tl
2284        }
2285      \int_gzero:N \g__enumext_check_starred_cmd_int
2286      \tl_clear:N \l__enumext_check_start_line_env_tl
2287    }
```

(*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.

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

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.

```
2288  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
2289    {
2290      \keys_define:nn { enumext / #1 }
2291        {
2292          mark-ans*   .tl_set:c  = { l__enumext_mark_answer_sym_#2_tl },
2293          mark-ans*   .initial:n = \textasteriskcentered,
```

```
2294            mark-ans*   .value_required:n = true,
2295            mark-pos*   .choice:,
2296            mark-pos* / left     .code:n = \str_set:cn { l__enumext_mark_position_#2_str } { l },
2297            mark-pos* / right    .code:n = \str_set:cn { l__enumext_mark_position_#2_str } { r },
2298            mark-pos* / center   .code:n = \str_set:cn { l__enumext_mark_position_#2_str } { c },
2299            mark-pos* / unknown .code:n =
2300                             \msg_error:nneee { enumext } { unknown-choice }
2301                                { mark-pos } { left,~right,~center } { \exp_not:n {##1} },
2302            mark-pos*   .initial:n = right,
2303            mark-pos*   .value_required:n = true,
2304            mark-sep*   .dim_set:c = { l__enumext_mark_sym_sep_#2_dim },
2305            mark-sep*   .value_required:n = true,
2306            wrap-ans*   .cs_set_protected:cp = { __enumext_keyans_wrapper_item_#2:n } ##1,
2307            wrap-ans*   .value_required:n = true,
2308            wrap-opt    .cs_set_protected:cp = { __enumext_keyans_wrapper_opt_#2:n } ##1,
2309            wrap-opt    .initial:n = [{##1}],
2310            wrap-opt    .value_required:n = true,
2311            save-sep    .tl_set:c = { l__enumext_store_keyans_item_opt_sep_#2_tl },
2312            save-sep    .initial:n = {,~},
2313            save-sep    .value_required:n = true,
2314            show-ans    .bool_set:N = \l__enumext_show_answer_bool,
2315            show-ans    .initial:n = false,
2316            show-ans    .value_required:n = true,
2317            show-pos    .bool_set:N = \l__enumext_show_position_bool,
2318            show-pos    .initial:n = false,
2319            show-pos    .value_required:n = true,
2320          }
2321      }
2322   \clist_map_inline:nn { {keyans}{v}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }
```

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

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

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*.

```
2323   \cs_set_protected:Npn \__enumext_tmp:n #1
2324     {
2325        \keys_define:nn { enumext / #1 }
2326          {
2327            mark-ref   .tl_set:N   = \l__enumext_mark_ref_sym_tl,
2328            mark-ref   .initial:n  = \textreferencemark,
2329            mark-ref   .value_required:n = true,
2330            save-ref   .bool_set:N = \l__enumext_store_ref_key_bool,
2331            save-ref   .initial:n  = false,
2332            save-ref   .value_required:n = true,
2333            show-ans   .bool_set:N = \l__enumext_show_answer_bool,
2334            show-ans   .initial:n  = false,
2335            show-ans   .value_required:n = true,
2336            show-pos   .bool_set:N = \l__enumext_show_position_bool,
2337            show-pos   .initial:n  = false,
2338            show-pos   .value_required:n = true,
2339            mark-ans   .tl_set:N   = \l__enumext_mark_answer_sym_tl,
2340            mark-ans   .initial:n = \textasteriskcentered,
2341            mark-ans   .value_required:n = true,
2342            mark-sep   .dim_set:N  = \l__enumext_mark_sym_sep_dim,
2343            mark-sep   .value_required:n = true,
2344            mark-pos   .choice:,
2345            mark-pos / left     .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2346            mark-pos / right    .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2347            mark-pos / center   .code:n = \str_set:Nn \l__enumext_mark_position_str { c },
2348            mark-pos / unknown .code:n =
2349                             \msg_error:nneee { enumext } { unknown-choice }
2350                                { mark-pos } { left,~right,~center } { \exp_not:n {##1} },
2351            mark-pos   .initial:n = right,
2352            mark-pos   .value_required:n = true,
2353
2354            wrap-ans   .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
2355            wrap-ans   .initial:n =
2356                          {
2357                             \fbox{\parbox[t]{\dimeval{\itemwidth -2\fboxsep -2\fboxrule}}{##1}}
```

```
2358                           },
2359          wrap-ans  .value_required:n = true,
2360          mark-ans* .code:n = {
2361                                 \keys_set:nn { enumext / keyans  } { mark-ans* = {##1} }
2362                                 \keys_set:nn { enumext / keyans* } { mark-ans* = {##1} }
2363                               },
2364          mark-ans* .value_required:n = true,
2365          mark-pos* .code:n = {
2366                                 \keys_set:nn { enumext / keyans  } { mark-pos* = {##1} }
2367                                 \keys_set:nn { enumext / keyans* } { mark-pos* = {##1} }
2368                               },
2369          mark-pos* .value_required:n = true,
2370          mark-sep* .code:n = {
2371                                 \keys_set:nn { enumext / keyans  } { mark-sep* = {##1} }
2372                                 \keys_set:nn { enumext / keyans* } { mark-sep* = {##1} }
2373                               },
2374          mark-sep* .value_required:n = true,
2375          wrap-ans* .code:n = {
2376                                 \keys_set:nn { enumext / keyans  } { wrap-ans* = {##1} }
2377                                 \keys_set:nn { enumext / keyans* } { wrap-ans* = {##1} }
2378                               },
2379          wrap-ans* .value_required:n = true,
2380          wrap-opt  .code:n = {
2381                                 \keys_set:nn { enumext / keyans  } { wrap-opt = {##1} }
2382                                 \keys_set:nn { enumext / keyans* } { wrap-opt = {##1} }
2383                               },
2384          wrap-opt  .value_required:n = true,
2385          save-sep  .code:n = {
2386                                 \keys_set:nn { enumext / keyans  } { save-sep = {##1} }
2387                                 \keys_set:nn { enumext / keyans* } { save-sep = {##1} }
2388                               },
2389          save-sep  .value_required:n = true,
2390        }
2391    }
2392 \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*⟩.

```
2393 \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2394   {
2395     \bool_if:cF { l__enumext_store_save_key_ \__enumext_level: _bool }
2396       {
2397         \tl_clear:c { l__enumext_store_save_key_ \__enumext_level: _tl }
2398         \tl_set:ce
2399           { l__enumext_store_save_key_ \__enumext_level: _tl }
2400           { \__enumext_filter_save_key:n {#1} }
2401       }
2402   }
2403 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2404   {
2405     \bool_if:NF \l__enumext_store_save_key_vii_bool
2406       {
2407         \tl_clear:N \l__enumext_store_save_key_vii_tl
2408         \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2409       }
2410   }
```

(*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.

```
2411 \cs_set_protected:Npn \__enumext_tmp:n #1
2412   {
2413     \keys_define:nn { enumext / enumext* }
2414       {
2415         save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2416         save-key .value_required:n = true,
2417       }
2418     \keys_define:nn { enumext / #1 }
2419       {
2420         save-key .code:n = \__enumext_parse_save_key:n {##1},
2421         save-key .value_required:n = true,
2422       }
2423   }
2424 \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*.

```
2425 \cs_new_protected:Npn \__enumext_parse_save_key:n #1
2426   {
2427     \bool_set_true:c { l__enumext_store_save_key_ \__enumext_level: _bool }
2428     \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2429     \tl_set:ce
2430       { l__enumext_store_save_key_ \__enumext_level: _tl }
2431       { \__enumext_filter_save_key:n {#1} }
2432   }
2433 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2434   {
2435     \bool_set_true:N \l__enumext_store_save_key_vii_bool
2436     \tl_clear:N \l__enumext_store_save_key_vii_tl
2437     \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2438   }
```

(*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

```
2439 \cs_new:Npn \__enumext_filter_save_key:n #1
2440   {
2441     \use:e
2442       {
2443         \keyval_parse:NNn
2444           \__enumext_filter_save_key_key:n
2445           \__enumext_filter_save_key_pair:nn {#1}
2446       }
2447   }
```

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.

```
2448 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2449   {
2450     \str_case:nnF {#1}
2451       {
2452         { resume } {} { resume* } {} { no-store } {} { base-fix } {}
2453       }
2454       { , { \exp_not:n {#1} } }
2455   }
```

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.

```
2456  \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2457    {
2458      \str_case:nnF {#1}
2459        {
2460          { series   } {} { resume    } {} { save-ans   } {} { save-ref  } {}
2461          { save-key } {} { check-ans } {} { show-ans   } {} { show-pos  } {}
2462          { mark-ans } {} { mark-pos  } {} { mark-sep   } {} { wrap-ans  } {}
2463          { mark-ans* } {} { mark-pos* } {} { mark-sep*  } {} { wrap-ans* } {}
2464          { wrap-opt } {} { save-sep  } {} { mark-ref   } {} { mini-env  } {}
2465          { mini-sep } {} { mini-right } {} { mini-right* } {}
2466        }
2467        { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2468    }
```

(*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.

```
2469  \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2470    {
2471      \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2472        {
2473          \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2474        }
2475        { #1 }
2476    }
2477  \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.

```
2478  \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2479    {
2480      \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
2481    }
2482  \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.

```
2483  \cs_new_protected:Nn \__enumext_store_level_open:
2484    {
2485      \bool_if:NT \l__enumext_check_answers_bool
2486        {
2487          \tl_if_empty:cTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2488            {
2489              \__enumext_store_addto_seq:n
2490                {
2491                  \item \begin{enumext}
2492                }
2493            }
2494            {
2495              \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
```

```
2496                    {
2497                        \item \begin{enumext} [
2498                    }
2499                \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2500                    {
2501                        ]
2502                    }
2503                \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2504            }
2505        }
2506    }
2507 \cs_new_protected:Nn \__enumext_store_level_close:
2508    {
2509        \bool_if:NT \l__enumext_check_answers_bool
2510            {
2511                \__enumext_store_addto_seq:n { \end{enumext} }
2512            }
2513    }
```

(*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.

```
2514 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2515    {
2516        \bool_if:NT \l__enumext_check_answers_bool
2517            {
2518                \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2519                    {
2520                        \__enumext_store_addto_seq:n
2521                            {
2522                                \item \begin{enumext*}
2523                            }
2524                    }
2525                    {
2526                        \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2527                            {
2528                                \item \begin{enumext*}[
2529                            }
2530                        \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2531                            {
2532                                ]
2533                            }
2534                        \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2535                    }
2536            }
2537    }
2538 \cs_new_protected:Nn \__enumext_store_level_close_vii:
2539    {
2540        \bool_if:NT \l__enumext_check_answers_bool
2541            {
2542                \__enumext_store_addto_seq:n { \end{enumext*} }
2543            }
2544    }
```

(*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

```
2545 \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2546    {
2547        \mode_leave_vertical:
2548        \skip_horizontal:n { -\dim_use:N #2 }
2549        \hbox_overlap_left:n
2550            {
2551                \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2552                    {
```

```
2553                    \tl_use:N \l__enumext_mark_answer_sym_tl
2554                }
2555            }
2556        \skip_horizontal:n { \dim_use:N #2 }
2557    }
2558 \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.

```
2559 \cs_new_protected:Nn \__enumext_store_internal_ref:
2560    {
2561        \cs_set_protected:Npn \__enumext_tmp:n ##1
2562            {
2563                \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2564                \tl_reverse:c { l__enumext_label_copy_##1_tl }
2565                \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2566                \tl_reverse:c { l__enumext_label_copy_##1_tl }
2567            }
2568        \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2569        \cs_set:Npn \__enumext_tmp:n ##1
2570            { . \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.

```
2571        \bool_lazy_all:nT
2572            {
2573                { \bool_if_p:N \g__enumext_starred_bool }
2574                { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2575            }
2576            {
2577                \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2578                    { \tl_use:N \l__enumext_label_copy_vii_tl }
2579            }
2580        \bool_lazy_all:nT
2581            {
2582                { \bool_not_p:n { \g__enumext_standar_bool } }
2583                { \bool_if_p:N \l__enumext_standar_bool }
2584                { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2585            }
2586            {
2587                \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2588                    {
2589                        \tl_use:N \l__enumext_label_copy_vii_tl
2590                        \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2591                    }
2592            }
```

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.

```
2593        \bool_lazy_all:nT
2594            {
2595                { \bool_if_p:N \g__enumext_standar_bool }
2596                { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2597                { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2598            }
2599            {
2600                \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2601                    {
2602                        \tl_use:N \l__enumext_label_copy_i_tl
2603                        \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2604                    }
2605            }
2606        \cs_set:Npn \__enumext_tmp:n ##1
2607            { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } . }
```

```
2608      \bool_lazy_all:nT
2609        {
2610          { \bool_if_p:N \g__enumext_standar_bool }
2611          { \bool_if_p:N \l__enumext_starred_bool }
2612          { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2613        }
2614        {
2615          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2616            {
2617              \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2618              \tl_use:N \l__enumext_label_copy_vii_tl
2619            }
2620        }
```

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

```
2621      \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2622        {
2623          \l__enumext_store_name_tl \c_colon_str
2624          \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2625        }
```

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.

```
2626      \tl_put_right:Ne \l__enumext_write_aux_file_tl
2627        {
2628          \__enumext_newlabel:nn
2629            { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2630            { \l__enumext_newlabel_arg_two_tl }
2631        }
2632    \l__enumext_write_aux_file_tl
2633  }
```

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

13.29   Common functions for \anskey and anskey* environment

`\__enumext_store_anskey_arg:n` The internal function `\__enumext_store_anskey_arg: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*⟩}.

```
2634 \cs_new_protected:Npn \__enumext_store_anskey_arg:n #1
2635   {
2636     \int_gincr:N \g__enumext_item_anskey_int
2637     \__enumext_store_addto_prop:n {#1}
2638     \bool_if:NT \l__enumext_store_ref_key_bool
2639       {
2640         \__enumext_store_internal_ref:
2641       }
2642     \__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`.

```
2643      \tl_clear:N \l__enumext_store_anskey_arg_tl
2644      \bool_lazy_and:nnT
2645        { \bool_if_p:N \l__enumext_store_columns_break_bool }
2646        { \bool_not_p:n { \l__enumext_starred_bool } }
2647        {
2648          \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2649        }
2650      \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`.

```
2651      \bool_lazy_and:nnT
2652        { \bool_not_p:n { \l__enumext_starred_bool } }
2653        { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2654        {
2655          \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2656            {
2657              ( \exp_not:V \l__enumext_store_item_join_int )
2658            }
```

```
2659          }
```

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*`.

```
2660        \bool_if:NTF \l__enumext_store_item_star_bool
2661          {
2662            \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2663            \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2664              {
2665                \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2666                  {
2667                    [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2668                  }
2669              }
2670            \dim_compare:nT
2671              {
2672                \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2673              }
2674              {
2675                \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2676                  {
2677                    [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2678                  }
2679              }
2680            \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2681          }
2682          {
2683            \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2684          }
```

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*.

```
2685        \bool_lazy_and:nnT
2686          { \bool_if_p:N \l__enumext_store_ref_key_bool }
2687          { \bool_if_p:N \l__enumext_hyperref_bool }
2688          {
2689            \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2690              {
2691                \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2692                  { \exp_not:V \l__enumext_mark_ref_sym_tl }
2693              }
2694          }
2695        \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2696      }
```

(*End of definition for* `\__enumext_store_anskey_arg: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.

```
2697  \cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
2698    {
2699      \par
2700      \bool_if:NTF \l__enumext_starred_bool
2701        {
2702          \dim_compare:nNnT { \l__enumext_mark_sym_sep_dim } = { \c_zero_dim }
2703            {
2704              \dim_set:Nn \l__enumext_mark_sym_sep_dim { \l__enumext_labelsep_vii_dim }
2705            }
2706          \__enumext_print_keyans_box:NN
2707            \l__enumext_labelwidth_vii_dim \l__enumext_mark_sym_sep_dim
2708        }
2709        {
2710          \dim_compare:nNnT { \l__enumext_mark_sym_sep_dim } = { \c_zero_dim }
2711            {
2712              \dim_set:Nn \l__enumext_mark_sym_sep_dim
2713                {
2714                  \dim_use:c {l__enumext_labelsep_ \__enumext_level: _dim }
2715                }
2716            }
2717          \__enumext_print_keyans_box:cc
2718            { l__enumext_labelwidth_ \__enumext_level: _dim } { l__enumext_mark_sym_sep_dim }
2719        }
```

```
2720        \__enumext_anskey_wrapper:n { #1 }
2721      }
```

(*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.

```
2722  \cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
2723    {
2724      \bool_if:NT \l__enumext_show_answer_bool
2725        {
2726          \__enumext_anskey_show_wrap_arg:n { #1 }
2727        }
2728      \bool_if:NT \l__enumext_show_position_bool
2729        {
2730          \tl_set:Ne \l__enumext_mark_answer_sym_tl
2731            {
2732              \group_begin:
2733              \exp_not:N \normalfont
2734              \exp_not:N \footnotesize [ \int_eval:n
2735                {
2736                  \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2737                }
2738                ]
2739              \group_end:
2740            }
2741          \__enumext_anskey_show_wrap_arg:n { #1 }
2742        }
2743    }
```

(*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
item-join
item-star
item-sym*
item-pos*
unknown
__enumext_anskey_unknown:n
__enumext_anskey_unknown:nn

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

```
2744  \keys_define:nn { enumext / anskey }
2745    {
2746      break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2747      break-col .default:n  = true,
2748      break-col .value_forbidden:n = true,
2749      item-join .int_set:N  = \l__enumext_store_item_join_int,
2750      item-join .value_required:n = true,
2751      item-star .bool_set:N = \l__enumext_store_item_star_bool,
2752      item-star .default:n  = true,
2753      item-star .value_forbidden:n = true,
2754      item-sym* .tl_set:N   = \l__enumext_store_item_symbol_tl,
2755      item-sym* .value_required:n = true,
2756      item-pos* .dim_set:N  = \l__enumext_store_item_symbol_sep_dim,
2757      item-pos* .value_required:n = true,
2758      unknown   .code:n     = { \__enumext_anskey_unknown:n {#1} },
2759    }
```

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.

```
2760  \cs_new_protected:Npn \__enumext_anskey_unknown:n #1
2761    {
2762      \exp_args:NV \__enumext_anskey_unknown:nn \l_keys_key_str {#1}
2763    }
2764  \cs_new_protected:Npn \__enumext_anskey_unknown:nn #1 #2
2765    {
2766      \tl_if_blank:nTF {#2}
2767        {
2768          \msg_error:nnn { enumext } { anskey-cmd-key-unknown } {#1}
2769        }
2770        {
```

```
2771            \msg_error:nnnn { enumext } { anskey-cmd-key-value-unknown } {#1} {#2}
2772          }
2773      }
```

(*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_arg:n.

```
2774  \NewDocumentCommand \anskey { o +m }
2775    {
2776      \__enumext_anskey_safe_outer:
2777      \group_begin:
2778        \bool_if:NT \l__enumext_check_answers_bool
2779          {
2780            \tl_if_novalue:nF {#1}
2781              {
2782                \keys_set:nn { enumext / anskey } {#1}
2783              }
2784            \tl_if_blank:nTF {#2}
2785              {
2786                \msg_error:nn { enumext } { anskey-empty-arg }
2787              }
2788              {
2789                \__enumext_anskey_safe_inner:
2790                \__enumext_store_anskey_arg:n {#2}
2791              }
2792          }
2793      \group_end:
2794    }
```

(*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.

```
2795  \cs_new_protected:Nn \__enumext_anskey_safe_outer:
2796    {
2797      \bool_if:NF \l__enumext_store_active_bool
2798        {
2799          \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
2800        }
2801      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2802        {
2803          \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
2804        }
2805      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2806        {
2807          \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans* }
2808        }
2809      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2810        {
2811          \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
2812        }
2813    }
```

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.

```
2814  \cs_new_protected:Nn \__enumext_anskey_safe_inner:
2815    {
2816      \int_incr:N \l__enumext_anskey_level_int
2817      \int_compare:nNnT { \l__enumext_anskey_level_int } > { 1 }
2818        {
2819          \msg_error:nn { enumext } { anskey-nested }
2820        }
2821      \bool_if:NF \l__enumext_item_number_bool
```

```
2822        {
2823            \msg_error:nn { enumext } { anskey-unnumber-item }
2824        }
2825    \mode_if_math:T
2826        {
2827            \msg_error:nne { enumext } { anskey-math-mode } { \c_backslash_str anskey }
2828        }
2829    }
```

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

13.31   The environment anskey*

The original implementation of the anskey* environment used non-public functions from the scontents package, which was not the best approach. Fortunately LaTeX release 2025-06-01 implemented the new c argument in the ltcmd module, with which we can record the ⟨*body*⟩ of the environment in *verbatim mode* and, together with \scantokens do the work as the original implementation.

break-col
item-join
item-star
item-sym*
item-pos*
force-eol
write-env
overwrite
unknown

First we add the same keys from the \anskey command along with the force-eol, write-env and overwrite keys that were in the original implementation that used the scontents support package for these.

```
2830 \keys_define:nn { enumext / anskey* }
2831    {
2832        break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2833        break-col .default:n  = true,
2834        break-col .value_forbidden:n = true,
2835        item-join .int_set:N  = \l__enumext_store_item_join_int,
2836        item-join .value_required:n = true,
2837        item-star .bool_set:N = \l__enumext_store_item_star_bool,
2838        item-star .default:n  = true,
2839        item-star .value_forbidden:n = true,
2840        item-sym* .tl_set:N   = \l__enumext_store_item_symbol_tl,
2841        item-sym* .value_required:n = true,
2842        item-pos* .dim_set:N  = \l__enumext_store_item_symbol_sep_dim,
2843        item-pos* .value_required:n = true,
2844        force-eol .bool_set:N = \l__enumext_anskey_env_force_eol_bool,
2845        force-eol .initial:n  = false,
2846        force-eol .default:n  = true,
2847        write-env .code:n     = {
2848                                  \bool_set_true:N \l__enumext_write_anskey_env_bool
2849                                  \tl_set:Nn \l__enumext_write_anskey_env_file_name_tl {#1}
2850                                },
2851        write-env .value_required:n = true,
2852        overwrite .bool_set:N = \l__enumext_anskey_env_overwrite_bool,
2853        overwrite .initial:n  = false,
2854        overwrite .default:n  = true,
2855        unknown   .code:n     = { \__enumext_anskey_env_unknown:n {#1} },
2856    }
```

(*End of definition for* break-col *and others.*)

__enumext_anskey_env_unknown:n
__enumext_anskey_env_unknown:nn

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.

```
2857 \cs_new_protected:Npn \__enumext_anskey_env_unknown:n #1
2858    {
2859        \exp_args:NV \__enumext_anskey_env_unknown:nn \l_keys_key_str {#1}
2860    }
2861 \cs_new_protected:Npn \__enumext_anskey_env_unknown:nn #1#2
2862    {
2863        \tl_if_blank:nTF {#2}
2864            {
2865                \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
2866            }
2867            {
2868                \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
2869            }
2870    }
```

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

The conditional function `\__enumext_anskey_env_file_if_writable:n` used by the `write-env` and `overwrite` keys in the `anskey*` environment to determine whether the output file is written or overwritten.

```
2871  \prg_new_protected_conditional:Npnn \__enumext_anskey_env_file_if_writable:n #1 { T, F, TF }
2872    {
2873      \bool_if:NTF \l__enumext_write_anskey_env_bool
2874        {
2875          \file_if_exist:nTF {#1}
2876            {
2877              \bool_if:NTF \l__enumext_anskey_env_overwrite_bool
2878                {
2879                  \msg_warning:nne { enumext } { overwrite-file } {#1}
2880                  \prg_return_true:
2881                }
2882                {
2883                  \msg_warning:nne { enumext } { not-writing } {#1}
2884                  \prg_return_false:
2885                }
2886            }
2887            {
2888              \msg_warning:nne { enumext } { writing-file } {#1}
2889              \prg_return_true:
2890            }
2891        }
2892        { \prg_return_false: }
2893    }
```

The `\__enumext_anskey_env_file_write:nn` function is used by the `write-env` key in the `anskey*` environment to write the output file with the ⟨*body*⟩ of the environment.

```
2894  \cs_new_protected:Npn \__enumext_anskey_env_file_write:nn #1#2
2895    {
2896      \__enumext_anskey_env_file_if_writable:nT {#1}
2897        {
2898          \iow_open:Nn \l__enumext_write_anskey_env_file_iow {#1}
2899          \iow_now:Nn  \l__enumext_write_anskey_env_file_iow {#2}
2900          \iow_close:N \l__enumext_write_anskey_env_file_iow
2901        }
2902    }
2903  \cs_generate_variant:Nn \__enumext_anskey_env_file_write:nn { VV }
```

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

First, we'll call the function `\__enumext_anskey_env_safe_outer:` to make sure where we're running the environment, then, we'll check the state of the variable `\l__enumext_check_answers_bool` set by the key `no-store`. If it's true, we'll look for [⟨*key = val*⟩] and verify that the *argument* c ⟨*body*⟩ is not empty. Finally, we'll run the internal check function `\__enumext_anskey_env_safe_inner:n` and call the function `\__enumext_store_anskey_arg:n`.

```
2904  \NewDocumentEnvironment{anskey*}{ o c }
2905    {
2906      \__enumext_anskey_env_safe_outer:
2907      \bool_if:NT \l__enumext_check_answers_bool
2908        {
2909          \tl_if_novalue:nF {#1}
2910            {
2911              \keys_set:nn { enumext / anskey* } {#1}
2912            }
2913          \tl_if_blank:nTF {#2}
2914            {
2915              \msg_error:nn { enumext } { anskey-empty-arg }
2916            }
2917            {
2918              \__enumext_anskey_env_safe_inner:
2919              \__enumext_store_anskey_env:n {#2}
2920            }
2921        }
2922    } { }
```

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

13.31.1   Internal functions for the environment

The function __enumext_store_anskey_safe_outer: will return the appropriate messages when anskey* is executed outside the environment in which the save-ans key was activated or within the keyans, keyans* or keyanspic environments.

```
2923  \cs_new_protected:Nn \__enumext_anskey_env_safe_outer:
2924    {
2925      \bool_if:NF \l__enumext_store_active_bool
2926        {
2927          \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2928        }
2929      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2930        {
2931          \msg_error:nnn { enumext } { anskey-env-wrong }{ keyans }
2932        }
2933      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2934        {
2935          \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
2936        }
2937      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2938        {
2939          \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
2940        }
2941    }
```

The function __enumext_anskey_env_safe_inner: will first check if preceded by a not numbered \item or if it is in *math mode* returning the appropriate messages.

```
2942  \cs_new_protected:Nn \__enumext_anskey_env_safe_inner:
2943    {
2944      \bool_if:NF \l__enumext_item_number_bool
2945        {
2946          \msg_error:nn { enumext } { anskey-unnumber-item }
2947        }
2948      \mode_if_math:T
2949        {
2950          \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
2951        }
2952    }
```

The __enumext_store_anskey_env:n function will first pass the argument c ⟨*body*⟩ to the variable \l__enumext_store_anskey_env_tl and replace the macro \obeyedline with ^^J and then execute the write-env and overwrite keys, check the state of the variable \l__enumext_anskey_env_force_eol_-bool managed by the force-eol key and we will add \c__enumext_anskey_env_hidden_space_str if necessary. Finally we will use \exp_args:Ne on the __enumext_store_anskey_arg:n to expand the __enumext_scan_tokens:n function which rescans the \l__enumext_store_anskey_env_tl variable before processing it.

```
2953  \cs_new_protected:Npn \__enumext_store_anskey_env:n #1
2954    {
2955      \tl_set:Nn \l__enumext_store_anskey_env_tl {#1}
2956      \RenewDocumentCommand \obeyedline { } { \iow_char:N \^^J }
2957      \tl_replace_all:Nee \l__enumext_store_anskey_env_tl { \obeyedline } { \iow_char:N \^^J }
2958      \__enumext_anskey_env_file_write:VV
2959        \l__enumext_write_anskey_env_file_name_tl \l__enumext_store_anskey_env_tl
2960      \bool_if:NF \l__enumext_anskey_env_force_eol_bool
2961        {
2962          \tl_put_right:Ne \l__enumext_store_anskey_env_tl
2963            {
2964              \c__enumext_anskey_env_hidden_space_str
2965            }
2966        }
2967      \exp_args:Ne
2968        \__enumext_store_anskey_arg:n
2969          {
2970            \__enumext_scan_tokens:n { \l__enumext_store_anskey_env_tl }
2971          }
2972    }
```

🖉 Since \obeyedline can be redefined by the user, for example to \mbox{}\par, it is necessary to redefine it to ^^J in order to use \tl_replace_all:Nee otherwise it returns an error.

(*End of definition for* __enumext_anskey_env_safe_outer: , __enumext_anskey_env_safe_inner: , *and* __enumext_store_-anskey_env:n.)

13.32 Executing check-ans system 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 and finally we execute the function __enumext_reset_global_vars: returning the used variables to their original state.

```
2973 \cs_new_protected:Nn \__enumext_execute_after_env:
2974   {
2975     \int_compare:nNnT { \l__enumext_level_int } = { 0 }
2976       {
2977         \tl_if_empty:NF \g__enumext_store_name_tl
2978           {
2979             \__enumext_stop_save_ans_msg:
2980             \__enumext_item_answer_diff:
2981             \__enumext_log_global_vars:
2982             \__enumext_log_answer_vars:
2983             \bool_if:NTF \g__enumext_check_ans_key_bool
2984               {
2985                 \__enumext_check_ans_show:
2986               }
2987               { \__enumext_check_ans_log: }
2988           }
2989         \__enumext_reset_global_vars:
2990       }
2991   }
```

🖈 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..

(*End of definition for* __enumext_execute_after_env:.)

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.

```
2992 \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2993   {
2994     \tl_clear:N \l__enumext_store_current_label_tl
2995     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2996       {
2997         \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
2998       }
2999       {
3000         \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
3001       }
```

If the *optional argument* is present and the save-sep key is not empty, we save it.

```
3002     \tl_if_novalue:nF { #1 }
3003       {
3004         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_v_tl
3005           {
3006             \tl_put_right:Nn \l__enumext_store_current_label_tl
3007               {
3008                 \l__enumext_store_keyans_item_opt_sep_v_tl
3009               }
3010           }
3011         \tl_put_right:Nn \l__enumext_store_current_label_tl { #1 }
3012       }
3013     \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
3014   }
```

(*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.

```
3015  \cs_new_protected:Nn \__enumext_keyans_store_ref:
3016    {
3017      \bool_if:NT \l__enumext_store_ref_key_bool
3018        {
3019          \cs_set_protected:Npn \__enumext_tmp:n ##1
3020            {
3021              \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
3022              \tl_reverse:c { l__enumext_label_copy_##1_tl }
3023              \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
3024              \tl_reverse:c { l__enumext_label_copy_##1_tl }
3025            }
3026          \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
3027          \__enumext_keyans_store_ref_aux_i:
3028        }
3029    }
```

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`.

```
3030  \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
3031    {
3032      \bool_if:NT \g__enumext_starred_bool
3033        {
3034          \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
3035        }
3036      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
3037        {
3038          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
3039            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
3040        }
3041      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
3042        {
3043          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
3044            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
3045        }
3046      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
3047        {
3048          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
3049            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
3050        }
3051      \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
3052        {
3053          \l__enumext_store_name_tl \c_colon_str
3054          \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
3055        }
3056      \__enumext_keyans_store_ref_aux_ii:
3057    }
```

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.

```
3058  \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
3059    {
3060      \tl_put_right:Ne \l__enumext_write_aux_file_tl
3061        {
3062          \__enumext_newlabel:nn
3063            { \exp_not:V \l__enumext_newlabel_arg_one_tl }
3064            { \l__enumext_newlabel_arg_two_tl }
3065        }
3066      \l__enumext_write_aux_file_tl
3067    }
```

(*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.

```
3068  \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
3069    {
3070      \tl_clear:N \l__enumext_store_current_label_tl
3071      \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
3072        {
3073          \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
3074        }
3075        {
3076          \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
3077        }
3078      \tl_if_novalue:nF { #1 }
3079        {
3080          \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_v_tl
3081            {
3082              \tl_put_right:Nn \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_o
3083            }
3084          \tl_put_right:Nn \l__enumext_store_current_label_tl { #1 }
3085        }
3086      \__enumext_keyans_addto_seq_link:
3087    }
```

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.

```
3088  \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
3089    {
3090      \bool_lazy_and:nnT
3091        { \bool_if_p:N \l__enumext_store_ref_key_bool }
3092        { \bool_if_p:N \l__enumext_hyperref_bool }
3093        {
3094          \tl_put_right:Ne \l__enumext_store_current_label_tl
3095            {
3096              \hfill \exp_not:N \hyperlink
3097                {
3098                  \exp_not:V \l__enumext_newlabel_arg_one_tl
3099                }
3100                { \exp_not:V \l__enumext_mark_ref_sym_tl }
3101            }
3102        }
3103      \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
3104      \bool_if:NT \l__enumext_check_answers_bool
3105        {
3106          \int_gincr:N \g__enumext_item_anskey_int
3107        }
3108    }
```

(*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.

```
3109  \cs_new_protected:Npn \__enumext_keyans_save_item_opt:n #1
3110    {
3111      \tl_if_novalue:nF { #1 }
3112        {
3113          \tl_set:Nn \l__enumext_store_current_opt_arg_tl { #1 }
3114        }
3115    }
```

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.

```
3116  \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
3117    {
3118      \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
3119        {
3120          \bool_lazy_or:nnT
3121            { \bool_if_p:N \l__enumext_show_answer_bool }
3122            { \bool_if_p:N \l__enumext_show_position_bool }
3123            {
3124              \__enumext_keyans_wrapper_opt_v:n
3125                { \l__enumext_store_current_opt_arg_tl } \c_space_tl
3126            }
3127        }
3128    }
```

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.

```
3129  \cs_new_protected:Nn \__enumext_keyans_show_item_opt_viii:
3130    {
3131      \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
3132        {
3133          \bool_lazy_or:nnT
3134            { \bool_if_p:N \l__enumext_show_answer_bool }
3135            { \bool_if_p:N \l__enumext_show_position_bool }
3136            {
3137              \__enumext_keyans_wrapper_opt_viii:n
3138                { \l__enumext_store_current_opt_arg_tl } \c_space_tl
3139            }
3140        }
3141    }
```

(*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:
__enumext_keyans_show_ans:
__enumext_keyans_show_pos:

The function __enumext_keyans_pos_mark_set: adjusts the horizontal spaces for the mark-sep* key
taking into account the value of the align key and the width of ⟨label⟩.

```
3142  \cs_new_protected:Nn \__enumext_keyans_pos_mark_set:
3143    {
3144      \__enumext_label_width_by_box:Nn
3145        \l__enumext_mark_sep_tmpa_dim { \l__enumext_label_v_tl }
3146      \str_case:Vn \l__enumext_align_label_pos_v_str
3147        {
3148          { l }
3149            {
3150              \dim_set:Nn \l__enumext_mark_sep_tmpb_dim { \c_zero_dim }
3151            }
3152          { r }
3153            {
3154              \dim_set:Nn \l__enumext_mark_sep_tmpb_dim
3155                { \l__enumext_labelwidth_v_dim - \l__enumext_mark_sep_tmpa_dim }
3156            }
3157          { c }
3158            {
3159              \dim_set:Nn \l__enumext_mark_sep_tmpb_dim
3160                { 0.5\l__enumext_labelwidth_v_dim - 0.5\l__enumext_mark_sep_tmpa_dim }
3161            }
3162        }
```

Here we set the default values for the key mark-ans*, mark-sep* and mark-pos*.

```
3163      \dim_compare:nNnT { \l__enumext_mark_sym_sep_v_dim } = { \c_zero_dim }
3164        {
3165          \dim_set:Nn \l__enumext_mark_sym_sep_v_dim { \l__enumext_labelsep_v_dim }
3166        }
3167      \tl_set_eq:NN \l__enumext_mark_answer_sym_tl \l__enumext_mark_answer_sym_v_tl
3168      \dim_add:Nn \l__enumext_mark_sym_sep_v_dim { \l__enumext_mark_sep_tmpb_dim }
3169      \str_set_eq:NN \l__enumext_mark_position_str \l__enumext_mark_position_v_str
3170    }
```

The function __enumext_keyans_show_ans: will print the ⟨symbol⟩ set by the mark-ans* key when the
show-ans key is active.

```
3171  \cs_new_protected:Nn \__enumext_keyans_show_ans:
3172    {
3173      \bool_lazy_all:nT
```

```
3174              {
3175                  { \bool_if_p:N \l__enumext_show_answer_bool   }
3176                  { \bool_if_p:N \l__enumext_item_wrap_key_bool }
3177              }
3178              {
3179                  \__enumext_keyans_pos_mark_set:
3180                  \__enumext_print_keyans_box:NN
3181                    \l__enumext_labelwidth_v_dim \l__enumext_mark_sym_sep_v_dim
3182              }
3183      }
```

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.

```
3184  \cs_new_protected:Nn \__enumext_keyans_show_pos:
3185    {
3186      \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
3187        {
3188          \int_incr:N \l__enumext_show_pos_tmp_int
3189        }
3190        {
3191          \int_zero:N \l__enumext_show_pos_tmp_int
3192        }
3193      \bool_lazy_all:nT
3194        {
3195          { \bool_if_p:N \l__enumext_show_position_bool }
3196          { \bool_if_p:N \l__enumext_item_wrap_key_bool }
3197        }
3198        {
3199          \tl_set:Ne \l__enumext_mark_answer_sym_v_tl
3200            {
3201              \group_begin:
3202                \exp_not:N \normalfont
3203                \exp_not:N \footnotesize [ \int_eval:n
3204                  {
3205                    \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3206                    + \l__enumext_show_pos_tmp_int
3207                  }
3208                ]
3209              \group_end:
3210            }
3211          \__enumext_keyans_pos_mark_set:
3212          \__enumext_print_keyans_box:NN
3213            \l__enumext_labelwidth_v_dim \l__enumext_mark_sym_sep_v_dim
3214        }
3215    }
```

(*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`.

```
3216  \cs_new_protected:Npn \__enumext_default_item:n #1
3217    {
3218      \tl_if_novalue:nTF {#1}
3219        {
```

```
3220        \bool_if:NT \l__enumext_check_answers_bool
3221          {
3222            \int_gincr:N \g__enumext_item_number_int
3223            \bool_set_true:N \l__enumext_item_number_bool
3224          }
3225        \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
3226        \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3227      }
3228      {
3229        \bool_set_eq:cc
3230          { l__enumext_wrap_label_ \__enumext_level: _bool }
3231          { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
3232        \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl
3233      }
3234    }
```

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

__enumext_item_starred_exec:nn
__enumext_item_starred_exec:

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

#1: \l__enumext_item_symbol_X_tl
#2: \l__enumext_item_symbol_sep_X_dim

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

```
3235  \cs_new_protected:Npn \__enumext_item_starred_exec:nn #1 #2
3236    {
3237      \tl_if_novalue:nTF {#1}
3238        {
3239          \tl_gset_eq:Nc
3240            \g__enumext_item_symbol_aux_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
3241        }
3242        {
3243          \tl_gset:Nn \g__enumext_item_symbol_aux_tl {#1}
3244        }
3245      \tl_if_novalue:nTF {#2}
3246        {
3247          \dim_set_eq:cc
3248            { l__enumext_item_symbol_sep_ \__enumext_level: _dim }
3249            { l__enumext_labelsep_ \__enumext_level: _dim }
3250        }
3251        {
3252          \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
3253        }
3254      \bool_if:NT \l__enumext_check_answers_bool
3255        {
3256          \int_gincr:N \g__enumext_item_number_int
3257          \bool_set_true:N \l__enumext_item_number_bool
3258        }
3259      \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
3260      \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3261    }
```

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

```
3262  \cs_new_protected:Nn \__enumext_item_starred_exec:
3263    {
3264      \tl_if_empty:cF { l__enumext_item_symbol_ \__enumext_level: _tl }
3265        {
3266          \mode_leave_vertical:
3267          \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3268          \hbox_overlap_left:n { \g__enumext_item_symbol_aux_tl }
3269          \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3270        }
3271    }
```

(*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).

```
3272 \cs_new_protected:Nn \__enumext_redefine_item:
3273   {
3274     \RenewDocumentCommand \item { s o o }
3275       {
3276         \bool_if:nTF {##1}
3277           {
3278             \__enumext_item_starred_exec:nn {##2} {##3}
3279           }
3280           { \__enumext_default_item:n {##2} }
3281       }
3282   }
```

(*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).

```
3283 \cs_new_protected:Nn \__enumext_make_label:
3284   {
3285     \IfDocumentMetadataTF
3286       {
3287         \__enumext_make_label_box:
3288       }
3289       {
3290         \bool_if:NTF \l__enumext_mode_box_bool
3291           {
3292             \__enumext_make_label_box:
3293           }
3294           {
3295             \__enumext_make_label_std:
3296           }
3297       }
3298   }
```

Standard definition when `\DocumentMetadata` is not active.

```
3299 \cs_new_protected:Nn \__enumext_make_label_std:
3300   {
3301     \RenewDocumentCommand \makelabel { m }
3302       {
3303         \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
3304         \__enumext_item_starred_exec:
3305         \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3306         \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3307           {
3308             \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3309           }
3310           { ##1 }
3311         \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
3312         \tl_gclear:N \g__enumext_item_symbol_aux_tl
3313       }
3314   }
```

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.

```
3315 \cs_new_protected:Nn \__enumext_make_label_box:
3316   {
3317     \RenewDocumentCommand \makelabel { m }
3318       {
3319         \strut\smash
3320           {
3321             \makebox
3322               [ \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim } ]
3323               [ \str_use:c { l__enumext_align_label_pos_ \__enumext_level: _str } ]
3324               {
3325                 \__enumext_item_starred_exec:
```

```
3326                  \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3327                  \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3328                    {
3329                      \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3330                    }
3331                    { ##1 }
3332                  \tl_gclear:N \g__enumext_item_symbol_aux_tl
3333                }
3334            } % close smash
3335        }
3336    }
```

(*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*
```
3337 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
3338    {
3339      \keys_define:nn { enumext / #1 }
3340        {
3341          item-sym* .tl_set:c  = { l__enumext_item_symbol_#2_tl },
3342          item-sym* .value_required:n = true,
3343          item-sym* .initial:n  = {\textborn},
3344          item-pos* .dim_set:c  = { l__enumext_item_symbol_sep_#2_dim },
3345          item-pos* .value_required:n = true,
3346        }
3347    }
3348 \clist_map_inline:nn
3349    {
3350      {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
3351    }
3352    { \__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   Define and set unknown key for keyans, keyans* and keyanspic environments. Here it is necessary to set
__enumext_keyans_unknown_keys:n   \l__enumext_envir_name_tl in case an unknown key is passed using \setenumext.
__enumext_keyans_unknown_keys:nn
```
3353 \cs_set_protected:Npn \__enumext_tmp:n #1
3354    {
3355      \keys_define:nn { enumext / #1 }
3356        {
3357          unknown .code:n = {
3358                            \tl_set:Nn \l__enumext_envir_name_tl {#1}
3359                            \__enumext_keyans_unknown_keys:n {##1}
3360                          },
3361        }
3362    }
3363 \clist_map_inline:nn { keyans, keyans*, keyanspic } { \__enumext_tmp:n {#1} }
```

Internal functions for handling unknown key.

```
3364 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:n #1
3365    {
3366      \exp_args:NV \__enumext_keyans_unknown_keys:nn \l_keys_key_str {#1}
3367    }
3368 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:nn #1#2
3369    {
3370      \tl_if_blank:nTF {#2}
3371        {
3372          \msg_error:nne { enumext } { keyans-unknown-key } {#1}
```

```
3373          }
3374          {
3375              \msg_error:nnee { enumext } { keyans-unknown-key-value } {#1} {#2}
3376          }
3377      }
```

(*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.

```
3378  \keys_define:nn { enumext / enumext* }
3379      {
3380          unknown .code:n = { \__enumext_starred_unknown_keys:n {#1} },
3381      }
```

Internal functions for handling unknown key.

```
3382  \cs_new_protected:Npn \__enumext_starred_unknown_keys:n #1
3383      {
3384          \exp_args:NV \__enumext_starred_unknown_keys:nn \l_keys_key_str {#1}
3385      }
3386  \cs_new_protected:Npn \__enumext_starred_unknown_keys:nn #1#2
3387      {
3388          \tl_if_blank:nTF {#2}
3389          {
3390              \msg_error:nnn { enumext } { starred-unknown-key } {#1}
3391          }
3392          {
3393              \msg_error:nnnn { enumext } { starred-unknown-key-value } {#1} {#2}
3394          }
3395      }
```

(*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.

```
3396  \cs_set_protected:Npn \__enumext_tmp:n #1
3397      {
3398          \keys_define:nn { enumext / #1 }
3399          {
3400              unknown .code:n = { \__enumext_standar_unknown_keys:n {##1} },
3401          }
3402      }
3403  \clist_map_inline:nn { level-1,level-2,level-3,level-4 } { \__enumext_tmp:n {#1} }
```

Internal functions for handling unknown key.

```
3404  \cs_new_protected:Npn \__enumext_standar_unknown_keys:n #1
3405      {
3406          \exp_args:NV \__enumext_standar_unknown_keys:nn \l_keys_key_str {#1}
3407      }
3408  \cs_new_protected:Npn \__enumext_standar_unknown_keys:nn #1#2
3409      {
3410          \tl_if_blank:nTF {#2}
3411          {
3412              \msg_error:nnn { enumext } { standar-unknown-key } {#1}
3413          }
3414          {
3415              \msg_error:nnnn { enumext } { standar-unknown-key-value } {#1} {#2}
3416          }
3417      }
```

(*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.

```
3418  \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
3419      {
```

```
3420      \tl_if_novalue:nTF { #1 }
3421        {
3422          \bool_set_true:N \l__enumext_wrap_label_v_bool
3423          \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
3424        }
3425        {
3426          \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
3427          \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
3428        }
3429      }
```

(*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.

```
3430  \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
3431    {
3432      \__enumext_keyans_save_item_opt:n { #1 }
3433      \bool_set_true:N \l__enumext_wrap_label_v_bool
3434      \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
3435      \__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.

```
3436      \__enumext_keyans_addto_prop:n { #1 }
3437      \__enumext_keyans_store_ref:
3438      \__enumext_keyans_addto_seq:n { #1 }
3439      \int_gincr:N \g__enumext_check_starred_cmd_int
3440    }
```

(*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).

```
3441  \cs_new_protected:Nn \__enumext_keyans_redefine_item:
3442    {
3443      \RenewDocumentCommand \item { s o }
3444        {
3445          \bool_if:nTF {##1}
3446            {
3447              \bool_set_true:N \l__enumext_item_wrap_key_bool % wrap-ans*
3448              \peek_remove_spaces:n
3449                {
3450                  \__enumext_keyans_starred_item:n {##2}
3451                }
3452            }
3453            {
3454              \bool_set_false:N \l__enumext_item_wrap_key_bool
3455              \__enumext_keyans_default_item:n {##2}
3456            }
3457        }
3458    }
```

(*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).

```
3459  \cs_new_protected:Nn \__enumext_keyans_make_label:
3460    {
3461      \IfDocumentMetadataTF
3462        {
3463          \__enumext_keyans_make_label_box:
```

```
3464        }
3465        {
3466            \bool_if:NTF \l__enumext_mode_box_bool
3467                {
3468                    \__enumext_keyans_make_label_box:
3469                }
3470                {
3471                    \__enumext_keyans_make_label_std:
3472                }
3473        }
3474    }
```

We added conditionals to the __enumext_keyans_wraper_label:n function to handle the keys wrap-ans*, wrap-label and wrap-label*.

```
3475 \cs_new_protected:Npn \__enumext_keyans_wrapper_label:n #1
3476    {
3477        \bool_lazy_all:nT
3478            {
3479                { \bool_if_p:N \l__enumext_wrap_label_v_bool          }
3480                { \bool_if_p:N \l__enumext_show_answer_bool           }
3481                { \bool_if_p:N \l__enumext_item_wrap_key_bool         }
3482                { \cs_if_exist_p:N \__enumext_keyans_wrapper_item_v:n }
3483            }
3484            {
3485                \cs_set_eq:NN \__enumext_wrapper_label_v:n \__enumext_keyans_wrapper_item_v:n
3486            }
3487        \bool_if:NTF \l__enumext_wrap_label_v_bool
3488            {
3489                \__enumext_wrapper_label_v:n { #1 }
3490            }
3491            { #1 }
3492    }
```

Standard definition when \DocumentMetadata is not active.

```
3493 \cs_new_protected:Nn \__enumext_keyans_make_label_std:
3494    {
3495        \RenewDocumentCommand \makelabel { m }
3496            {
3497                \tl_use:N \l__enumext_label_fill_left_v_tl
3498                \__enumext_keyans_show_ans:
3499                \__enumext_keyans_show_pos:
3500                \tl_use:N \l__enumext_label_font_style_v_tl
3501                \__enumext_keyans_wrapper_label:n { ##1 }
3502                \tl_use:N \l__enumext_label_fill_right_v_tl
3503            }
3504    }
```

Definition using \makebox when \DocumentMetadata is active or mode-box is active.

```
3505 \cs_new_protected:Nn \__enumext_keyans_make_label_box:
3506    {
3507        \RenewDocumentCommand \makelabel { m }
3508            {
3509                \strut\smash
3510                    {
3511                        \makebox[ \l__enumext_labelwidth_v_dim ][ \l__enumext_align_label_pos_v_str ]
3512                            {
3513                                \__enumext_keyans_show_ans:
3514                                \__enumext_keyans_show_pos:
3515                                \tl_use:N \l__enumext_label_font_style_v_tl
3516                                \__enumext_keyans_wrapper_label:n { ##1 }
3517                            }
3518                    }
3519            }
3520    }
```

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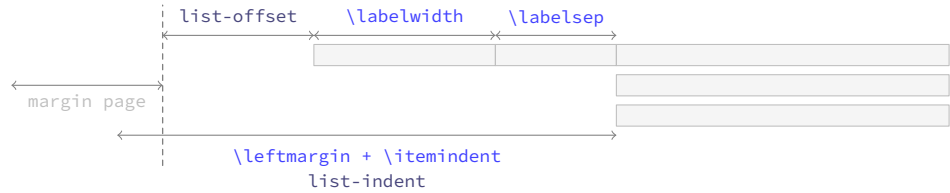
Figure 9: Representation of standard horizontal lengths in `list` environment.

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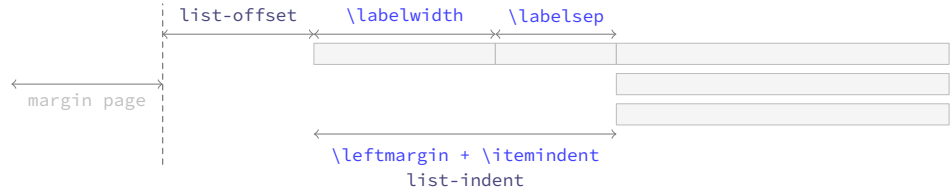
Figure 10: Representation of horizontal lengths concept in list in enumext.

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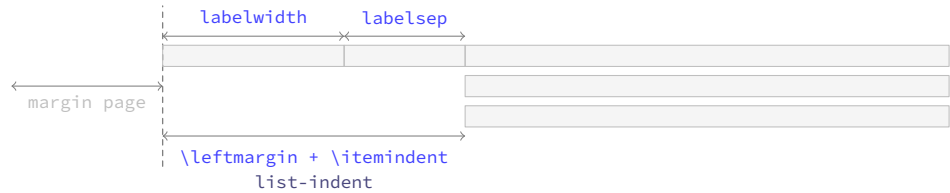
Figure 11: Default horizontal lengths in enumext.

`\__enumext_calc_hspace:NNNNNNN`
`\__enumext_calc_hspace:ccccccc`

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

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

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

This function is passed to `\__enumext_list_arg_two_X:` which is used in the definition of the enumext and keyans environments (§13.38).

```
3521  \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
3522    {
3523      \dim_compare:nNnT { #1 } < { \c_zero_dim }
3524        {
3525          \msg_warning:nnnV { enumext } { width-non-positive }{ labelwidth }{ #1 }
3526          \dim_set:Nn #1 { \dim_abs:n { #1 } }
3527        }
3528      \dim_compare:nNnT { #2 } < { \c_zero_dim }
3529        {
3530          \msg_warning:nnnV { enumext } { width-negative }{ labelsep }{ #2 }
3531          \dim_set:Nn #2 { \dim_abs:n { #2 } }
3532        }
```

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

```
3533        \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2} }
```

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

```
3534      \dim_compare:nNnTF { #4 } < { \c_zero_dim }
3535        {
3536          \dim_set:Nn #6 { #1 + #2 - #4}
3537          \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3538        }
3539        {
3540          \dim_compare:nNnT { #4 } = { #1 + #2 }
```

```
3541            { \dim_set:Nn #6 { \c_zero_dim } }
3542        \dim_compare:nNnT { #4 } < { #1 + #2 }
3543            { \dim_set:Nn #6 { #1 + #2 - #4} }
3544        \dim_compare:nNnT { #4 } > { #1 + #2 }
3545            {
3546                \dim_set:Nn #6 { -#1 - #2 + #4}
3547                \dim_set:Nn #6 { #6*-1}
3548            }
3549        \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3550      }
3551    }
3552  \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.

```
3553  \cs_set_protected:Npn \__enumext_tmp:n #1
3554    {
3555      \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3556        {
3557          \__enumext_calc_hspace:ccccccc
3558            { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3559            { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3560            { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3561            { l__enumext_leftmargin_tmp_#1_bool }
3562          \clist_map_inline:nn
3563            { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3564            { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
3565          \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3566            { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
3567          \usecounter { enumX#1 }
3568          \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
3569          \str_if_eq:nnTF {#1} { v }
3570            {
3571              \__enumext_keyans_redefine_item:
3572              \__enumext_keyans_make_label:
3573              \__enumext_keyans_ref:
3574              \__enumext_keyans_fake_item_indent:
3575              \bool_if:cT { l__enumext_show_length_#1_bool }
3576                {
3577                  \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3578                }
3579            }
3580            {
3581              \__enumext_redefine_item:
3582              \__enumext_make_label:
3583              \__enumext_standar_ref:
3584              \__enumext_fake_item_indent:
3585              \bool_if:cT { l__enumext_show_length_#1_bool }
3586                {
3587                  \msg_term:nnne { enumext } { list-lengths } {#1}
3588                    { \int_use:N \l__enumext_level_int }
3589                }
3590            }
3591        }
3592    }
3593  \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.

```
3594  \cs_set_protected:Npn \__enumext_tmp:n #1
3595    {
3596      \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3597        {
3598          \bool_set_true:c { l__enumext_leftmargin_tmp_#1_bool }
```

```
3599            \dim_zero:c { l__enumext_leftmargin_tmp_#1_dim }
3600            \__enumext_calc_hspace:ccccccc
3601              { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3602              { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3603              { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3604              { l__enumext_leftmargin_tmp_#1_bool }
3605            \clist_map_inline:nn
3606              { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3607              { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
3608            \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3609              { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
3610            \skip_set_eq:Nc \parsep  { l__enumext_itemsep_#1_skip }
3611            \skip_zero:N \partopsep
3612            \usecounter { enumX#1 }
3613            \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
3614            \__enumext_starred_ref:
3615            \str_if_eq:nnTF {#1} { vii }
3616              {
3617                \__enumext_fake_item_indent_vii:
3618                \bool_if:cT { l__enumext_show_length_vii_bool }
3619                  { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3620              }
3621              {
3622                \__enumext_fake_item_indent_viii:
3623                \bool_if:cT { l__enumext_show_length_#1_bool }
3624                  { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3625              }
3626          }
3627      }
3628    \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"*.

```
3629  \cs_new_protected:Nn \__enumext_safe_exec:
3630    {
3631      \__enumext_is_not_nested:
3632      \__enumext_internal_mini_page:
3633      \int_incr:N \l__enumext_level_int
3634      \int_compare:nNnT { \l__enumext_level_int } > { 4 }
3635        { \msg_fatal:nn { enumext } { list-too-deep } }
3636      \bool_set_true:N \l__enumext_standar_bool
3637      \bool_set_false:N \l__enumext_starred_bool
3638      \__enumext_is_on_first_level:
3639    }
```

(*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.

```
3640  \cs_new_protected:Npn \__enumext_parse_keys:n #1
3641    {
3642      \tl_if_novalue:nF {#1}
3643        {
3644          \str_clear:N \l__enumext_series_str
3645          \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
3646            {
3647              \keys_set:nn { enumext / level-1 } {#1}
3648              \__enumext_parse_series:n {#1}
3649              \__enumext_nested_base_line_fix:
```

```
3650            }
3651          {
3652            \exp_args:Ne \keys_set:nn
3653              { enumext / level-\int_use:N \l__enumext_level_int } {#1}
3654          }
3655        \__enumext_store_active_keys:n {#1}
3656      }
3657  }
```

(*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*.

```
3658  \cs_new_protected:Nn \__enumext_start_store_level:
3659    {
3660      \bool_lazy_all:nT
3661        {
3662          { \bool_if_p:N \l__enumext_store_active_bool }
3663          { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3664          { \bool_if_p:N \g__enumext_standar_bool }
3665        }
3666        {
3667          \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3668            {
3669              \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3670              \__enumext_store_level_open:
3671            }
3672        }
```

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

```
3673      \bool_lazy_all:nT
3674        {
3675          { \bool_if_p:N \l__enumext_store_active_bool }
3676          { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3677          { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
3678        }
3679        {
3680          \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3681            {
3682              \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3683              \__enumext_store_level_open:
3684            }
3685        }
3686  }
```

(*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*.

```
3687  \cs_new_protected:Nn \__enumext_stop_store_level:
3688    {
3689      \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
3690        {
3691          \__enumext_store_level_close:
3692        }
3693  }
```

(*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.

```
3694  \cs_new_protected:Nn \__enumext_multicols_start:
3695    {
3696      \int_compare:nNnT
3697        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3698        {
3699          \dim_compare:nNnT
3700            { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3701            {
3702              \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
```

```
3703                    {
3704                      ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
3705                        + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
3706                      ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
3707                      - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3708                    }
3709                  }
3710              \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
3711              \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3712                {
3713                  \dim_zero:N \columnseprule
3714                }
```

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.

```
3715              \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3716                {
3717                  \skip_zero:N \multicolsep
3718                  \__enumext_multi_addvspace:
3719                }
3720              \raggedcolumns
3721              \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
3722            }
3723        }
```

(*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:`.

```
3724  \cs_new_protected:Nn \__enumext_multicols_stop:
3725    {
3726      \int_compare:nNnTF
3727        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3728        {
3729          \__enumext_stop_list:
3730          \__enumext_stop_store_level:
3731          \end{multicols}
3732          \__enumext_unskip_unkern:
3733          \__enumext_unskip_unkern:
3734          \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
3735        }
3736        {
3737          \__enumext_stop_list:
3738          \__enumext_stop_store_level:
3739        }
3740    }
```

(*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.

```
3741  \cs_new_protected:Nn \__enumext_before_list:
3742    {
3743      \__enumext_vspace_above:
3744      \__enumext_before_args_exec:
3745      \__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 the `\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.

```
3746      \dim_compare:nNnT
3747        { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3748        {
3749          \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
3750            {
3751              \linewidth
3752              - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
```

```
3753                    - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
3754                }
```

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.

```
3755                \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
3756                \int_gincr:N \g__enumext_minipage_stat_int
3757                \__enumext_minipage_add_space:
3758                \noindent
3759                \__enumext_mini_page{ \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
3760            }
3761        \__enumext_multicols_start:
3762    }
```

(*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.

```
3763 \cs_new_protected:Nn \__enumext_second_part:
3764    {
3765        \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
3766            {
3767                \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3768                    {
3769                        \msg_warning:nn { enumext } { missing-miniright }
3770                        \miniright
3771                    }
3772                \int_gzero:N \g__enumext_minipage_stat_int
3773                \__enumext_unskip_unkern: % remove topsep + [partopsep]
3774                \end__enumext_mini_page
3775            }
3776            {
3777                \__enumext_multicols_stop:
3778            }
```

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.

```
3779        \__enumext_after_stop_list:
3780        \__enumext_check_ans_key_hook:
3781        \__enumext_vspace_below:
3782        \bool_set_false:N \l__enumext_standar_bool
3783        \__enumext_resume_save_counter:
3784    }
```

(*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.

```
3785 \cs_new_protected:Nn \__enumext_set_item_width:
3786    {
3787        \dim_set:Nn \itemwidth { \linewidth }
3788        \dim_compare:nT
3789            {
3790                \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim } != \c_zero_dim
3791            }
3792            {
3793                \dim_sub:Nn \itemwidth
3794                    {
3795                        \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3796                    }
3797            }
3798    }
```

(*End of definition for* __enumext_set_item_width:.)

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

```
3799 \NewDocumentEnvironment{enumext}{ O{} }
3800   {
3801     \__enumext_safe_exec:
3802     \__enumext_parse_keys:n {#1}
3803     \__enumext_before_list:
3804     \__enumext_start_store_level:
3805     \__enumext_start_list:nn
3806       { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
3807       {
3808         \use:c { __enumext_list_arg_two_ \__enumext_level: : }
3809         \__enumext_before_keys_exec:
3810       }
3811     \__enumext_set_item_width:
3812     \__enumext_after_args_exec:
3813   }
3814   {
3815     \__enumext_second_part:
3816   }
```

(*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.

```
3817 \__enumext_after_env:nn {enumext}
3818   {
3819     \__enumext_execute_after_env:
3820   }
```

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.

```
3821 \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3822   {
3823     \bool_if:NF \l__enumext_store_active_bool
3824       {
3825         \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
3826       }
3827     \int_incr:N \l__enumext_keyans_level_int
3828     \bool_set_true:N \l__enumext_keyans_env_bool
3829     \__enumext_keyans_name_and_start:
3830     % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3831     \bool_set_false:N \l__enumext_store_active_bool
3832     \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
3833       {
3834         \msg_error:nn { enumext } { keyans-nested }
3835       }
3836     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3837       {
3838         \msg_error:nn { enumext } { keyans-wrong-level }
3839       }
3840   }
```

(*End of definition for* __enumext_keyans_safe_exec:.)

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

```
3841 \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3842   {
3843     \keys_set:nn { enumext / keyans } {#1}
3844   }
```

(*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.

```
3845 \cs_new_protected:Nn \__enumext_before_list_v:
3846   {
3847     \__enumext_vspace_above_v:
3848     \__enumext_before_args_exec_v:
3849     \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3850       {
3851         \dim_set:Nn \l__enumext_minipage_left_v_dim
3852           {
3853             \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3854           }
3855         \bool_set_true:N \l__enumext_minipage_active_v_bool
3856         \int_gincr:N \g__enumext_minipage_stat_int
3857         \__enumext_keyans_minipage_add_space:
3858         \__enumext_mini_page{ \l__enumext_minipage_left_v_dim }
3859       }
3860     \__enumext_keyans_multicols_start:
3861   }
3862 \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3863   {
3864     \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3865       {
3866         \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3867           {
3868             \dim_set:Nn \l__enumext_columns_sep_v_dim
3869               {
3870                 (
3871                   \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3872                 ) / \l__enumext_columns_v_int
3873                 - \l__enumext_listoffset_v_dim
3874               }
3875           }
3876         \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim
3877         \dim_zero:N \columnseprule % no rule here
3878         \bool_if:NF \l__enumext_minipage_active_v_bool
3879           {
3880             \skip_zero:N \multicolsep
3881             \__enumext_keyans_multi_addvspace:
3882           }
3883         \raggedcolumns
3884         \begin{multicols}{ \l__enumext_columns_v_int }
3885       }
3886   }
3887 \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3888   {
3889     \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
3890       {
3891         \__enumext_stop_list:
3892         \end{multicols}
3893         \__enumext_unskip_unkern:
3894         \__enumext_unskip_unkern:
3895         \par\addvspace{ \l__enumext_multicols_below_v_skip }
3896       }
3897       {
3898         \__enumext_stop_list:
3899       }
3900   }
3901 \cs_new_protected:Nn \__enumext_second_part_v:
3902   {
3903     \bool_if:NTF \l__enumext_minipage_active_v_bool
3904       {
3905         \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3906           {
3907             \msg_warning:nn { enumext } { missing-miniright }
3908             \miniright
3909           }
3910         \int_gzero:N \g__enumext_minipage_stat_int
3911         \__enumext_unskip_unkern: % remove \topsep + [\partopsep]
3912         \end__enumext_mini_page
3913         \par\addvspace{ \l__enumext_minipage_after_skip }
3914       }
```

```
3915        {
3916            \__enumext_keyans_multicols_stop:
3917        }
3918      \bool_set_false:N \l__enumext_keyans_env_bool
3919      \__enumext_after_stop_list_v:
3920      \__enumext_vspace_below_v:
3921    }
```

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

__enumext_keyans_set_item_width: The function __enumext_keyans_set_item_width: will set the value of \itemwidth taking into account the value established by the list-offset key.

```
3922  \cs_new_protected:Nn \__enumext_keyans_set_item_width:
3923    {
3924      \dim_set:Nn \itemwidth { \linewidth }
3925      \dim_compare:nT
3926        {
3927            \l__enumext_listoffset_v_dim != \c_zero_dim
3928        }
3929        {
3930            \dim_sub:Nn \itemwidth { \l__enumext_listoffset_v_dim }
3931        }
3932    }
```

(*End of definition for __enumext_keyans_set_item_width:.*)

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

```
3933  \NewDocumentEnvironment{keyans}{ O{} }
3934    {
3935      \__enumext_keyans_safe_exec:
3936      \__enumext_keyans_parse_keys:n {#1}
3937      \__enumext_before_list_v:
3938      \__enumext_start_list:nn
3939        { \tl_use:N \l__enumext_label_v_tl }
3940        {
3941            \__enumext_list_arg_two_v:
3942            \__enumext_before_keys_exec_v:
3943        }
3944      \__enumext_keyans_set_item_width:
3945      \__enumext_after_args_exec_v:
3946    }
3947    {
3948      \__enumext_check_starred_cmd:n { item }
3949      \__enumext_second_part_v:
3950    }
```

(*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

```
3951  \socket_new:nn {tagsupport/__enumext/starred}{ 1 }
3952  \socket_new_plug:nnn {tagsupport/__enumext/starred} {start-list-tags}
3953    {
3954      \tag_resume:n {#1}
3955      \tag_mc_end_push:
3956        \tag_struct_begin:n {tag=LI}
3957          \tag_struct_begin:n {tag=Lbl}
3958            \tag_mc_begin:n {tag=Lbl}
3959    }
3960  \socket_new_plug:nnn {tagsupport/__enumext/starred} {stop-start-tags}
```

```
3961        {
3962                \tag_mc_end:
3963            \tag_struct_end:n {tag=Lbl}
3964            \tag_struct_begin:n {tag=LBody}
3965              \tag_struct_begin:n {tag=text-unit}
3966                \tag_struct_begin:n {tag=text}
3967        }
3968    \socket_new_plug:nnn {tagsupport/__enumext/starred} {stop-list-tags}
3969        {
3970                \tag_struct_end:n {tag=text}
3971              \tag_struct_end:n {tag=text-unit}
3972            \tag_struct_end:n {tag=LBody}
3973          \tag_struct_end:n {tag=LI}
3974        \tag_mc_begin_pop:n {}
3975        \tag_suspend:n {#1}
3976        }
```

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

```
3977    \cs_new_protected_nopar:Npn \__enumext_start_list_tag:n #1
3978        {
3979          \IfDocumentMetadataTF
3980            {
3981              \socket_assign_plug:nn {tagsupport/__enumext/starred} {start-list-tags}
3982              \socket_use:nn {tagsupport/__enumext/starred} {#1}
3983            } {}
3984        }
3985    \cs_new_protected_nopar:Nn \__enumext_stop_start_list_tag:
3986        {
3987          \IfDocumentMetadataTF
3988            {
3989              \socket_assign_plug:nn {tagsupport/__enumext/starred} {stop-start-tags}
3990              \socket_use:nn {tagsupport/__enumext/starred} { }
3991            } {}
3992        }
3993    \cs_new_protected_nopar:Npn \__enumext_stop_list_tag:n #1
3994        {
3995          \IfDocumentMetadataTF
3996            {
3997              \socket_assign_plug:nn {tagsupport/__enumext/starred} {stop-list-tags}
3998              \socket_use:nn {tagsupport/__enumext/starred} {#1}
3999            } {}
4000        }
```

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

13.41.2 Socket for tagging support in `keyanspic`

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

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:

```
4001    \socket_new:nn {tagsupport/__enumext/keyanspic}{ 0 }
4002    \socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {start-list-tags}
4003        {
4004          \tag_resume:n {keyanspic}
4005        \tag_mc_end_push:
4006          \tag_struct_begin:n {tag=LI}
4007            \tag_struct_begin:n {tag=Lbl}
4008              \tag_mc_begin:n {tag=Lbl}
4009        }
4010    \socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {stop-start-tags}
4011        {
4012                \tag_mc_end:
4013            \tag_struct_end:n {tag=Lbl}
4014            \tag_struct_begin:n {tag=LBody}
4015              \tag_struct_begin:n {tag=text-unit}
4016                \tag_struct_begin:n {tag=text}
4017                  \tag_mc_begin:n {tag=text}
4018        }
4019    \socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {stop-list-tags}
4020        {
4021                  \tag_mc_end:
4022              \tag_struct_end:n {tag=text}
4023              \tag_struct_end:n {tag=text-unit}
4024            \tag_struct_end:n {tag=LBody}
4025          \tag_struct_end:n {tag=LI}
```

```
4026        \tag_mc_begin_pop:n {}
4027        \tag_suspend:n {keyanspic}
4028      }
```

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

```
4029  \cs_new_protected_nopar:Nn \__enumext_anspic_start_list_tag:
4030    {
4031      \IfDocumentMetadataTF
4032        {
4033          \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {start-list-tags}
4034          \socket_use:n {tagsupport/__enumext/keyanspic}
4035        } {}
4036    }
4037  \cs_new_protected_nopar:Nn \__enumext_anspic_stop_start_list_tag:
4038    {
4039      \IfDocumentMetadataTF
4040        {
4041          \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {stop-start-tags}
4042          \socket_use:n {tagsupport/__enumext/keyanspic}
4043        } {}
4044    }
4045  \cs_new_protected_nopar:Nn \__enumext_anspic_stop_list_tag:
4046    {
4047      \IfDocumentMetadataTF
4048        {
4049          \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {stop-list-tags}
4050          \socket_use:n {tagsupport/__enumext/keyanspic}
4051        } {}
4052    }
```

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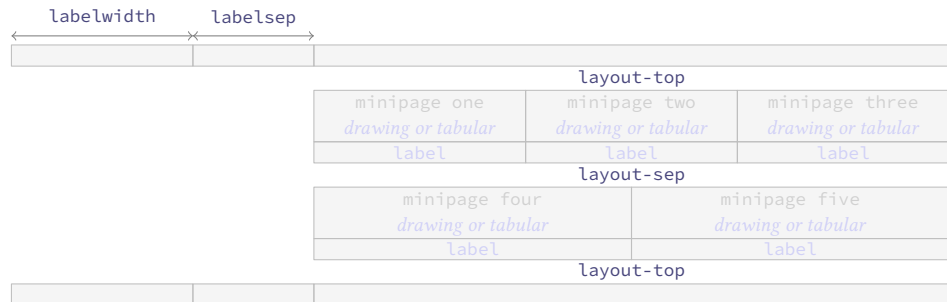
Figure 12: Representation of the `keyanspic` spacing in enumext.

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`.

```
4053  \keys_define:nn { enumext / keyanspic }
4054    {
4055      label-pos .choice:,
4056      label-pos / above   .code:n =
4057                          \bool_set_true:N \l__enumext_anspic_label_above_bool
4058                          \str_set:Nn \l__enumext_anspic_mini_pos_str { t },
4059      label-pos / below   .code:n =
```

```
4060                             \bool_set_false:N \l__enumext_anspic_label_above_bool
4061                             \str_set:Nn \l__enumext_anspic_mini_pos_str { b },
4062     label-pos / unknown .code:n =
4063                             \msg_error:nneee { enumext } { unknown-choice }
4064                               { label-pos } { above,~ below } { \exp_not:n {#1} },
4065     label-pos  .initial:n        = below,
4066     label-pos  .value_required:n = true,
4067     label-sep  .skip_set:N       = \l__enumext_anspic_label_sep_skip,
4068     label-sep  .value_required:n = true,
4069     layout-sty .tl_set:N         = \l__enumext_anspic_layout_style_tl,
4070     layout-sty .value_required:n = true,
4071     layout-sep .code:n           = \keys_set:nn { enumext / keyans } { parsep = #1 },
4072     layout-sep .value_required:n = true,
4073     layout-top .code:n           = \keys_set:nn { enumext / keyans } { topsep = #1 },
4074     layout-top .value_required:n = true,
4075     mark-ans   .code:n           = \keys_set:nn { enumext / keyans } { mark-ans = #1 },
4076     mark-ans   .value_required:n = true,
4077     mark-pos   .code:n           = \keys_set:nn { enumext / keyans } { mark-pos = #1 },
4078     mark-pos   .value_required:n = true,
4079     mark-sep   .code:n           = \keys_set:nn { enumext / keyans } { mark-sep = #1 },
4080     mark-sep   .value_required:n = true,
4081     save-sep   .code:n           = \keys_set:nn { enumext / keyans } { save-sep = #1 },
4082     save-sep   .value_required:n = true,
4083     wrap-opt   .code:n           = \keys_set:nn { enumext / keyans } { wrap-opt = #1 },
4084     wrap-opt   .value_required:n = true,
4085     wrap-ans*  .code:n           = \keys_set:nn { enumext / keyans } { wrap-ans* = #1 },
4086     wrap-ans*  .value_required:n = true,
4087     show-ans   .code:n           = \keys_set:nn { enumext / keyans } { show-ans = #1 },
4088     show-ans   .value_required:n = true,
4089     show-pos   .code:n           = \keys_set:nn { enumext / keyans } { show-pos = #1 },
4090     show-pos   .value_required:n = true,
4091     unknown    .code:n           = {
4092                             \tl_set:Nn \l__enumext_envir_name_tl { keyanspic }
4093                             \__enumext_keyans_unknown_keys:n {#1}
4094                               },
4095     }
```

(*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.

```
4096 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
4097   {
4098     \int_incr:N \l__enumext_keyans_pic_level_int
4099     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
4100       {
4101         \msg_error:nn { enumext } { keyanspic-nested }
4102       }
4103     \__enumext_keyans_name_and_start:
4104   }
```

Parse [⟨*key* = *val*⟩] for keyanspic environment.

```
4105 \cs_new_protected:Npn \__enumext_keyans_pic_parse_keys:n #1
4106   {
4107     \tl_if_novalue:nF {#1}
4108       {
4109         \keys_set:nn { enumext / keyanspic } {#1}
4110       }
4111   }
```

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

```
4112 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
4113   {
4114     \dim_compare:nNnT { #1 } < { \c_zero_dim }
4115       {
4116         \skip_set:Nn #1 { -#1 }
4117       }
4118   }
```

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.

```
4119  \cs_new_protected:Npn \__enumext_keyans_pic_arg_two:
4120    {
4121      \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
4122      \__enumext_list_arg_two_v:
4123      \__enumext_keyans_pic_skip_abs:N \parsep
```

Now we increment the counter enumXv of the `keyans` environment and save the *total height* of the ⟨*label*⟩ 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*.

```
4124      \bool_if:NF \l__enumext_anspic_label_above_bool
4125        {
4126          \stepcounter { enumXv }
4127          \hbox_set:Nn \l__enumext_anspic_label_box { \l__enumext_label_v_tl }
4128          \dim_set:Nn \l__enumext_anspic_label_htdp_dim
4129            {
4130              \box_ht_plus_dp:N \l__enumext_anspic_label_box
4131            }
4132          \skip_add:Nn \parsep
4133            {
4134              \l__enumext_anspic_label_htdp_dim
4135              + \box_dp:N \strutbox
4136              + \l__enumext_anspic_label_sep_skip
4137            }
4138        }
```

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.

```
4139      \dim_add:Nn \leftmargin { -\l__enumext_labelwidth_v_dim - \l__enumext_labelsep_v_dim }
4140      \ignorespaces
4141      \skip_add:Nn \topsep { 0.5\box_dp:N \strutbox }
4142      \dim_zero:N  \listparindent
4143      \skip_zero:N \partopsep
4144      \skip_zero:N \itemsep
4145    }
```

(*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.

```
4146  \NewDocumentEnvironment{keyanspic}{ o }
4147    {
4148      \__enumext_keyans_pic_safe_exec:
4149      \__enumext_keyans_pic_parse_keys:n {#1}
4150      \begin{list} { } { \__enumext_keyans_pic_arg_two: }
4151      \IfDocumentMetadataTF
4152        {
4153          \tag_suspend:n {list}
4154        }{}
4155      \item[] \scan_stop:
4156      \RenewDocumentCommand \item {}
4157        {
4158          \msg_error:nn { enumext } { keyanspic-item-cmd }
4159        }
4160      \IfDocumentMetadataTF
4161        {
4162          \tag_resume:n {keyanspic}
4163          \tag_tool:n {para/tagging=false}
4164          \tag_suspend:n {keyanspic}
4165        } { }
4166    }
```

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.

```
4167    {
4168      \IfDocumentMetadataTF
```

```
4169          {
4170            \tag_resume:n {keyanspic}
4171            \tag_mc_end_push:
4172            \tag_struct_begin:n {tag=L,attribute=enumerate}
4173          } { }
4174      \__enumext_anspic_exec:
4175      \IfDocumentMetadataTF
4176          {
4177            \tag_suspend:n {keyanspic}
4178          } { }
4179      \end{list}
4180      \IfDocumentMetadataTF
4181          {
4182            \tag_struct_end:n {tag=L}
4183            \tag_mc_begin_pop:n {}
4184            \tag_struct_end:n {tag=L}
4185            \tag_mc_begin_pop:n {}
4186          } { }
```

Finally we check if \anspic* has been used, set the counter enumXvi to zero and apply our "adjusted" vertical space bottom.

```
4187      \__enumext_check_starred_cmd:n { anspic }
4188      \setcounter { enumXvi } { 0 }
4189      \bool_if:NTF \l__enumext_anspic_label_above_bool
4190          {
4191            \par\addvspace{ 0.5\box_dp:N \strutbox }
4192          }
4193          {
4194            \par
4195            \addvspace
4196              {
4197                \dim_eval:n
4198                  {
4199                    \l__enumext_anspic_label_htdp_dim + \box_ht_plus_dp:N \strutbox
4200                    + \l__enumext_anspic_label_sep_skip + \l__enumext_topsep_v_skip
4201                  }
4202              }
4203          }
4204      }
```

(*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.

```
4205  \NewDocumentCommand \anspic { s o +m }
4206    {
4207      \bool_if:NF \l__enumext_store_active_bool
4208        {
4209          \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
4210        }
4211      \int_compare:nNnT { \l__enumext_level_int } > { 1 }
4212        {
4213          \msg_error:nn { enumext } { keyanspic-wrong-level }
4214        }
4215      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
4216        {
4217          \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
4218        }
4219      \seq_put_right:Nn \l__enumext_anspic_args_seq
4220        {
```

```
4221              \__enumext_anspic_args:nnn { #1 } { #2 } { #3 }
4222          }
4223      }
```

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".

```
4224  \cs_new_protected:Npn \__enumext_anspic_body_dim:n #1
4225    {
4226      \bool_if:NF \l__enumext_anspic_label_above_bool
4227        {
4228          \IfDocumentMetadataTF
4229            {
4230              \tag_suspend:n {keyanspic}
4231            } { }
4232          \vbox_set:Nn \l__enumext_anspic_body_box { #1 }
4233          \dim_set:Nn \l__enumext_anspic_body_htdp_dim
4234            {
4235              \box_ht_plus_dp:N \l__enumext_anspic_body_box
4236            }
4237          \IfDocumentMetadataTF
4238            {
4239              \tag_resume:n {keyanspic}
4240            } { }
4241        }
4242    }
```

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.

```
4243  \cs_new_protected:Npn \__enumext_anspic_label:nn #1 #2
4244    {
4245      \makebox[ \l__enumext_anspic_mini_width_dim ][ c ]
4246        {
4247          \bool_if:nTF { #1 }
4248            {
4249              \bool_set_true:N \l__enumext_item_wrap_key_bool
4250              \bool_set_true:N \l__enumext_wrap_label_v_bool
4251              \__enumext_keyans_save_item_opt:n { #2 }
4252              \__enumext_keyans_addto_prop:n { #2 }
4253              \__enumext_keyans_store_ref:
4254              \__enumext_keyans_addto_seq:n { #2 }
4255              \int_gincr:N \g__enumext_check_starred_cmd_int
4256              \__enumext_keyans_show_ans:
4257              \__enumext_keyans_show_pos:
4258              \makebox[ \l__enumext_labelwidth_v_dim ][c]
4259                {
4260                  \tl_use:N \l__enumext_label_font_style_v_tl
4261                  \__enumext_keyans_wrapper_label:n { \l__enumext_label_vi_tl }
4262                }
4263              \skip_horizontal:n { \l__enumext_labelsep_v_dim }
4264              \__enumext_keyans_show_item_opt:
4265            }
4266            {
4267              \bool_set_false:N \l__enumext_item_wrap_key_bool
4268              \tl_use:N \l__enumext_label_font_style_v_tl
4269              \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl }
4270            }
4271        }
4272    }
```

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.

```
4273  \cs_new_protected:Npn \__enumext_anspic_label_pos:nnn #1 #2 #3
4274    {
4275      \stepcounter { enumXvi }
4276      \__enumext_anspic_body_dim:n { #3 }
4277      \bool_if:NTF \l__enumext_anspic_label_above_bool
4278        {
4279          \__enumext_anspic_label:nn { #1 } { #2 }
4280        }
4281        {
4282          \raisebox
```

```
4283                    {
4284                      -\dim_eval:n
4285                        {
4286                          \l__enumext_anspic_label_htdp_dim
4287                          + \l__enumext_anspic_body_htdp_dim
4288                          + \box_dp:N \strutbox
4289                          + \l__enumext_anspic_label_sep_skip
4290                        }
4291                    }
4292                    [ 0pt ] [ 0pt ]
4293                    {
4294                      \__enumext_anspic_label:nn { #1 } { #2 }
4295                    }
4296                }
4297            }
4298  %
```

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.

```
4299  \cs_new_protected:Nn \__enumext_anspic_args:nnn
4300    {
4301      \__enumext_anspic_start_list_tag:
4302      \__enumext_anspic_label_pos:nnn { #1 } { #2 } { #3 }
4303      \__enumext_anspic_stop_start_list_tag:
4304      \bool_if:NTF \l__enumext_anspic_label_above_bool
4305        {
4306          \\[\l__enumext_anspic_label_sep_skip] #3
4307        }
4308        {
4309          \\ #3
4310        }
4311      \__enumext_anspic_stop_list_tag:
4312    }
```

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.

```
4313  \cs_new_protected:Nn \__enumext_anspic_print:n
4314    {
4315      \clist_map_function:nN { #1 } \__enumext_anspic_row:n
4316    }
4317  \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.

```
4318  \cs_new_protected:Nn \__enumext_anspic_row:n
4319    {
4320      \dim_set:Nn \l__enumext_anspic_mini_width_dim { \linewidth / #1 }
4321      \int_set:Nn \l__enumext_anspic_above_int { \l__enumext_anspic_below_int }
4322      \int_set:Nn \l__enumext_anspic_below_int { \l__enumext_anspic_above_int + #1 }
4323      \int_step_inline:nnn
4324        { \l__enumext_anspic_above_int + 1 }
4325        { \l__enumext_anspic_below_int }
4326        {
4327          \IfDocumentMetadataTF
4328            {
4329              \tag_suspend:n {minipage}
4330            } { }
4331          \begin{minipage}[ \l__enumext_anspic_mini_pos_str ]{ \l__enumext_anspic_mini_width_dim }
4332            \centering
4333            \seq_item:Nn \l__enumext_anspic_args_seq { ##1 }
4334          \end{minipage}
4335          \IfDocumentMetadataTF
4336            {
4337              \tag_resume:n {minipage}
4338            } { }
4339        }
4340      \par
4341    }
```

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*.

```
4342  \cs_new_protected:Nn \__enumext_anspic_exec:
4343    {
4344      \tl_if_empty:NTF \l__enumext_anspic_layout_style_tl
4345        {
4346          \__enumext_anspic_print:e { \seq_count:N \l__enumext_anspic_args_seq }
4347        }
4348        {
4349          \__enumext_anspic_print:V \l__enumext_anspic_layout_style_tl
4350        }
4351    }
```

(*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.

```
4352  \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
4353    {
4354      \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
4355        {
4356          \dim_set:Nn \l__enumext_columns_sep_vii_dim
4357            {
4358              ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
4359              / \l__enumext_columns_vii_int
4360            }
4361        }
4362      \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - 1 }
4363      \dim_set:Nn \l__enumext_item_width_vii_dim
4364        {
4365          ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
4366          / \l__enumext_columns_vii_int
4367          - \l__enumext_labelwidth_vii_dim
4368          - \l__enumext_labelsep_vii_dim
4369        }
```

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

```
4370      \dim_compare:nNnT { \l__enumext_rightmargin_vii_dim } > { \c_zero_dim }
4371        {
4372          \dim_sub:Nn \l__enumext_item_width_vii_dim
4373            {
4374              ( \l__enumext_rightmargin_vii_dim * \l__enumext_tmpa_vii_int )
4375              / \l__enumext_columns_vii_int
4376            }
4377          \dim_add:Nn \l__enumext_columns_sep_vii_dim
4378            {
4379              \l__enumext_rightmargin_vii_dim
4380            }
4381        }
4382    }
```

Same implementation for the keyans* environment.

```
4383  \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
4384    {
4385      \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
4386        {
4387          \dim_set:Nn \l__enumext_columns_sep_viii_dim
4388            {
4389              ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
4390              / \l__enumext_columns_viii_int
4391            }
4392        }
4393      \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - 1 }
4394      \dim_set:Nn \l__enumext_item_width_viii_dim
4395        {
4396          ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
4397          / \l__enumext_columns_viii_int
4398          - \l__enumext_labelwidth_viii_dim
4399          - \l__enumext_labelsep_viii_dim
4400        }
4401      \dim_compare:nNnT { \l__enumext_rightmargin_viii_dim } > { \c_zero_dim }
4402        {
4403          \dim_sub:Nn \l__enumext_item_width_viii_dim
4404            {
4405              ( \l__enumext_rightmargin_viii_dim * \l__enumext_tmpa_vii_int )
4406              / \l__enumext_columns_viii_int
4407            }
4408          \dim_add:Nn \l__enumext_columns_sep_viii_dim
4409            {
4410              \l__enumext_rightmargin_viii_dim
4411            }
4412        }
4413    }
```

(*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.

```
4414  \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
4415    {
4416      \int_set:Nn \l__enumext_joined_item_vii_int {#1}
4417      \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
4418        {
4419          \msg_warning:nnee { enumext } { item-joined }
4420            { \int_use:N \l__enumext_joined_item_vii_int }
4421            { \int_use:N \l__enumext_columns_vii_int }
4422          \int_set:Nn \l__enumext_joined_item_vii_int
4423            {
4424              \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4425            }
4426        }
4427      \int_compare:nNnT
4428        { \l__enumext_joined_item_vii_int }
4429        >
4430        { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4431        {
4432          \msg_warning:nnee { enumext } { item-joined-columns }
4433            { \int_use:N \l__enumext_joined_item_vii_int }
4434            {
4435              \int_eval:n
4436                { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4437            }
4438          \int_set:Nn \l__enumext_joined_item_vii_int
4439            {
4440              \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4441            }
4442        }
4443      \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { 1 }
4444        {
4445          \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
```

```
4446          \int_decr:N \l__enumext_joined_item_aux_vii_int
4447          \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
4448          \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
4449          \dim_set:Nn \l__enumext_joined_width_vii_dim
4450            {
4451              \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
4452              + (   \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
4453                + \l__enumext_columns_sep_vii_dim
4454              )*\l__enumext_joined_item_aux_vii_int
4455            }
4456          \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
4457        }
4458        {
4459          \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
4460          \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
4461        }
4462    }
```

Same implementation for the keyans* environment.

```
4463  \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
4464    {
4465      \int_set:Nn \l__enumext_joined_item_viii_int {#1}
4466      \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
4467        {
4468          \msg_warning:nnee { enumext } { item-joined }
4469            { \int_use:N \l__enumext_joined_item_viii_int }
4470            { \int_use:N \l__enumext_columns_viii_int }
4471          \int_set:Nn \l__enumext_joined_item_viii_int
4472            {
4473              \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4474            }
4475        }
4476      \int_compare:nNnT
4477        { \l__enumext_joined_item_viii_int }
4478          >
4479        { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4480        {
4481          \msg_warning:nnee { enumext } { item-joined-columns }
4482            { \int_use:N \l__enumext_joined_item_viii_int }
4483            {
4484              \int_eval:n
4485                { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4486            }
4487          \int_set:Nn \l__enumext_joined_item_viii_int
4488            {
4489              \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4490            }
4491        }
4492      \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { 1 }
4493        {
4494          \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
4495          \int_decr:N \l__enumext_joined_item_aux_viii_int
4496          \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
4497          \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
4498          \dim_set:Nn \l__enumext_joined_width_viii_dim
4499            {
4500              \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
4501              + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
4502                + \l__enumext_columns_sep_viii_dim
4503              )*\l__enumext_joined_item_aux_viii_int
4504            }
4505          \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
4506        }
4507        {
4508          \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
4509          \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
4510        }
4511    }
```

(*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`.

```
4512 \cs_new_protected:Nn \__enumext_start_mini_vii:
4513   {
4514     \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
4515       {
4516         \dim_set:Nn \l__enumext_minipage_left_vii_dim
4517           {
4518             \linewidth
4519             - \l__enumext_minipage_right_vii_dim
4520             - \l__enumext_minipage_hsep_vii_dim
4521           }
4522         \bool_set_true:N \l__enumext_minipage_active_vii_bool
4523         \dim_gset_eq:NN
4524           \g__enumext_minipage_right_vii_dim
4525           \l__enumext_minipage_right_vii_dim
4526         \__enumext_mini_addvspace_vii:
4527         \nointerlineskip\noindent
4528         \__enumext_mini_page{ \l__enumext_minipage_left_vii_dim }
4529       }
4530   }
```

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).

```
4531 \cs_new_protected:Nn \__enumext_stop_mini_vii:
4532   {
4533     \bool_if:NTF \l__enumext_minipage_active_vii_bool
4534       {
4535         \__enumext_stop_list:
4536         \__enumext_stop_store_level_vii:
4537         \IfDocumentMetadataTF { \tag_resume:n {enumext*} } { }
4538         \end__enumext_mini_page
4539         \hfill
4540         \bool_gset_true:N \g__enumext_minipage_active_vii_bool
4541       }
4542       {
4543         \__enumext_stop_list:
4544         \__enumext_stop_store_level_vii:
4545       }
4546   }
```

(*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.

```
4547 \__enumext_after_env:nn {enumext*}
4548   {
4549     \bool_if:NT \g__enumext_minipage_active_vii_bool
4550       {
4551         \__enumext_minipage:w [ t ] { \g__enumext_minipage_right_vii_dim }
4552         \legacy_if_gset_false:n { @minipage }
4553         \skip_vertical:N \c_zero_skip
4554         \par\addvspace { \g__enumext_minipage_right_skip }
4555         \bool_if:NF \g__enumext_minipage_center_vii_bool
4556           {
4557             \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
4558               {
4559                 \centering
4560               }
4561           }
4562         \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
```

```
4563              {
4564                \tl_use:N \g__enumext_miniright_code_vii_tl
4565              }
4566            \box_use_drop:N \l__enumext_miniright_code_vii_box
4567            \skip_vertical:N \c_zero_skip
4568          \__enumext_endminipage:
4569          \par\addvspace{ \g__enumext_minipage_after_skip }
4570        }
4571      \bool_gset_false:N \g__enumext_minipage_active_vii_bool
4572      \bool_gset_true:N \g__enumext_minipage_center_vii_bool
4573      \tl_gclear:N \g__enumext_miniright_code_vii_tl
4574      \dim_gzero:N \g__enumext_minipage_right_vii_dim
4575      \bool_gset_false:N \g__enumext_starred_bool
4576    }
```

`\__enumext_start_mini_viii:` The implementation of the `mini-env`, `mini-right` and `mini-right*` keys is identical to the one used in the
`\__enumext_stop_mini_viii:` `enumext*` environment.

```
4577  \cs_new_protected:Nn \__enumext_start_mini_viii:
4578    {
4579      \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
4580        {
4581          \dim_set:Nn \l__enumext_minipage_left_viii_dim
4582            {
4583              \linewidth
4584              - \l__enumext_minipage_right_viii_dim
4585              - \l__enumext_minipage_hsep_viii_dim
4586            }
4587          \bool_set_true:N \l__enumext_minipage_active_viii_bool
4588          \dim_gset_eq:NN
4589            \g__enumext_minipage_right_viii_dim
4590            \l__enumext_minipage_right_viii_dim
4591          \__enumext_mini_addvspace_viii:
4592          \nointerlineskip\noindent
4593          \__enumext_mini_page{ \l__enumext_minipage_left_viii_dim }
4594        }
4595    }
4596  \cs_new_protected:Nn \__enumext_stop_mini_viii:
4597    {
4598      \bool_if:NTF \l__enumext_minipage_active_viii_bool
4599        {
4600          \__enumext_stop_list:
4601          \IfDocumentMetadataTF { \tag_resume:n {keyans*} } { }
4602          \end__enumext_mini_page
4603          \hfill
4604          \bool_gset_true:N \g__enumext_minipage_active_viii_bool
4605        }
4606        {
4607          \__enumext_stop_list:
4608        }
4609    }
4610  \__enumext_after_env:nn {keyans*}
4611    {
4612      \bool_if:NT \g__enumext_minipage_active_viii_bool
4613        {
4614          \__enumext_mini_page{ \g__enumext_minipage_right_viii_dim }
4615            \par\addvspace { \g__enumext_minipage_right_skip }
4616            \bool_if:NF \g__enumext_minipage_center_viii_bool
4617              {
4618                \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
4619                  {
4620                    \centering
4621                  }
4622              }
4623            \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
4624              {
4625                \tl_use:N \g__enumext_miniright_code_viii_tl
4626              }
4627            \box_use_drop:N \l__enumext_miniright_code_viii_box
4628          \end__enumext_mini_page
4629          \par\addvspace{ \g__enumext_minipage_after_skip }
4630        }
```

```
4631    \bool_gset_false:N \g__enumext_minipage_active_viii_bool
4632    \bool_gset_true:N \g__enumext_minipage_center_viii_bool
4633    \tl_gclear:N \g__enumext_miniright_code_viii_tl
4634    \dim_gzero:N \g__enumext_minipage_right_viii_dim
4635  }
```

(*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.

```
4636  \NewDocumentEnvironment{enumext*}{ o }
4637    {
4638      \__enumext_safe_exec_vii:
4639      \__enumext_parse_keys_vii:n {#1}
4640      \__enumext_before_list_vii:
4641      \__enumext_start_store_level_vii:
4642      \__enumext_start_list:nn { }
4643        {
4644          \__enumext_list_arg_two_vii:
4645          \__enumext_before_keys_exec_vii:
4646        }
4647      \IfDocumentMetadataTF { \tag_suspend:n {enumext*} } { }
4648      \__enumext_starred_columns_set_vii:
4649      \item[] \scan_stop:
4650      \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_first_item_tmp_vii:
4651      \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
4652      \ignorespaces
4653    }
4654    {
4655      \IfDocumentMetadataTF { \tag_struct_end:n {tag=text-unit} } { }
4656      \__enumext_stop_item_tmp_vii:
4657      \__enumext_remove_extra_parsep_vii:
4658      \__enumext_after_list_vii:
4659    }
```

(*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.

```
4660  \cs_new_protected:Nn \__enumext_safe_exec_vii:
4661    {
4662      \__enumext_is_not_nested:
4663      \__enumext_internal_mini_page:
4664      \int_incr:N \l__enumext_level_h_int
4665      \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
4666        {
4667          \msg_error:nn { enumext } { nested }
4668        }
4669      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
4670        {
4671          \msg_error:nnn { enumext } { nested-horizontal } { keyans*}
4672        }
4673      \bool_set_true:N \l__enumext_starred_bool
4674      \bool_set_false:N \l__enumext_standar_bool
4675      \__enumext_is_on_first_level:
4676    }
```

(*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.

```
4677  \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
4678    {
4679      \tl_if_novalue:nF {#1}
4680        {
4681          \str_clear:N \l__enumext_series_str
4682          \keys_set:nn { enumext / enumext* } {#1}
4683          \__enumext_parse_series:n {#1}
4684          \__enumext_store_active_keys_vii:n {#1}
4685        }
4686    }
```

(*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*.

```
4687  \cs_new_protected:Nn \__enumext_before_list_vii:
4688    {
4689      \__enumext_vspace_above_vii:
4690      \__enumext_check_ans_active:
4691      \__enumext_before_args_exec_vii:
4692      \__enumext_start_mini_vii:
4693    }
```

(*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.

```
4694  \cs_new_protected:Nn \__enumext_after_list_vii:
4695    {
4696      \__enumext_stop_mini_vii:
4697      \__enumext_after_stop_list_vii:
4698      \__enumext_check_ans_key_hook:
4699      \__enumext_vspace_below_vii:
4700      \bool_set_false:N \l__enumext_starred_bool
4701      \__enumext_resume_save_counter:
4702    }
```

(*End of definition for* __enumext_after_list_vii:.)

__enumext_start_store_level_vii: The __enumext_start_store_level_vii: and __enumext_stop_store_level_vii: functions ac-
__enumext_stop_store_level_vii: tivate the *"storing structure"* mechanism in *sequence* for \anskey command and anskey* environment if enumext* are nested in enumext.

```
4703  \cs_new_protected:Nn \__enumext_start_store_level_vii:
4704    {
4705      \bool_if:NT \l__enumext_store_active_bool
4706        {
4707          \int_compare:nNnT { \l__enumext_level_int } > { 0 }
4708            {
4709              \__enumext_store_level_open_vii:
4710            }
4711        }
4712    }
4713  \cs_new_protected:Nn \__enumext_stop_store_level_vii:
4714    {
4715      \bool_if:NT \l__enumext_store_active_bool
4716        {
4717          \int_compare:nNnT { \l__enumext_level_int } > { 0 }
4718            {
4719              \__enumext_store_level_close_vii:
```

```
4720            }
4721        }
4722    }
```

(*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.

```
4723 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_vii:
4724    {
4725        \skip_horizontal:n
4726            {
4727                -\l__enumext_labelwidth_vii_dim - \l__enumext_labelsep_vii_dim
4728            }
4729        \ignorespaces
4730    }
```

(*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`.

```
4731 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
4732    {
4733        \__enumext_stop_item_tmp_vii:
4734        \int_incr:N \l__enumext_item_column_pos_vii_int
4735        \int_gincr:N \g__enumext_item_count_all_vii_int
4736        \__enumext_item_peek_args_vii:
4737    }
```

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`).

```
4738 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
4739    {
4740        \peek_meaning:NTF (
4741            { \__enumext_joined_item_vii:w }
4742            { \__enumext_joined_item_vii:w (1) }
4743    }
```

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`.

```
4744 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
4745    {
4746        \__enumext_starred_joined_item_vii:n {#1}
4747        \peek_meaning_remove:NTF *
4748            { \__enumext_starred_item_vii:w  }
4749            { \__enumext_standar_item_vii:w }
4750    }
```

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`].

```
4751 \cs_new_protected:Npn \__enumext_standar_item_vii:w
4752    {
4753        \bool_set_false:N \l__enumext_item_starred_vii_bool
4754        \peek_meaning:NTF [
4755            {
4756                \bool_set_eq:NN \l__enumext_wrap_label_vii_bool \l__enumext_wrap_label_opt_vii_bool
4757                \__enumext_start_item_vii:w
4758            }
```

```
4759                   {
4760                       \bool_set_true:N \l__enumext_wrap_label_vii_bool
4761                       \legacy_if_set_true:n { @noitemarg }
4762                       \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ] \ignorespaces
4763                   }
4764               }
```

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*⟩`]`.

```
4765   \cs_new_protected:Npn \__enumext_starred_item_vii:w
4766       {
4767           \bool_set_true:N \l__enumext_item_starred_vii_bool
4768           \bool_set_true:N \l__enumext_wrap_label_vii_bool
4769           \peek_meaning:NTF [
4770               { \__enumext_starred_item_vii_aux_i:w }
4771               { \__enumext_starred_item_vii_aux_ii:w }
4772       }
4773   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
4774       {
4775           \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
4776           \__enumext_starred_item_vii_aux_ii:w
4777       }
4778   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
4779       {
4780           \peek_meaning:NTF [
4781               { \__enumext_starred_item_vii_aux_iii:w }
4782               {
4783                   \dim_set_eq:NN \l__enumext_item_symbol_sep_vii_dim \l__enumext_labelsep_vii_dim
4784                   \legacy_if_set_true:n { @noitemarg }
4785                   \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ] \ignorespaces
4786               }
4787       }
4788   \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
4789       {
4790           \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
4791           \legacy_if_set_true:n { @noitemarg }
4792           \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ] \ignorespaces
4793       }
```

(*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`

```
4794   \cs_new_protected_nopar:Npn \__enumext_fake_make_label_vii:n #1
4795       {
4796           \legacy_if:nT { @noitemarg }
4797               {
4798                   \legacy_if_set_false:n { @noitemarg }
4799                   \legacy_if:nT { @nmbrlist }
4800                       {
4801                           \IfDocumentMetadataTF
4802                               {
4803                                   \bool_if:NT \l__enumext_hyperref_bool
4804                                       {
4805                                           \legacy_if_set_true:n { @hyper@item }
4806                                       }
4807                               } { }
4808                           \refstepcounter{enumXvii}
4809                           \bool_if:NT \l__enumext_check_answers_bool
4810                               {
4811                                   \int_gincr:N \g__enumext_item_number_int
```

```
4812                    \bool_set_true:N \l__enumext_item_number_bool
4813                  }
4814              }
4815          }
4816        \bool_if:NT \l__enumext_item_starred_vii_bool
4817          {
4818            \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
4819              {
4820                \tl_gset_eq:NN
4821                  \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
4822              }
4823            \mode_leave_vertical:
4824            \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
4825            \hbox_overlap_left:n { \g__enumext_item_symbol_aux_vii_tl }
4826            \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
4827            \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
4828          }
4829        \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
4830          {
4831            \tl_use:N \l__enumext_label_font_style_vii_tl
4832            \bool_if:NTF \l__enumext_wrap_label_vii_bool
4833              {
4834                \__enumext_wrapper_label_vii:n {#1}
4835              }
4836              { #1 }
4837          }
4838        \skip_horizontal:N \l__enumext_labelsep_vii_dim \ignorespaces
4839    }
```

(*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.

```
4840  \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
4841    {
4842      \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
4843      \hbox_set_to_wd:Nnw \l__enumext_item_text_vii_box
4844        {
4845          \l__enumext_joined_width_vii_dim
4846          + \l__enumext_labelwidth_vii_dim
4847          + \l__enumext_labelsep_vii_dim
4848        }
```

Redefine the \footnote command.

```
4849            \__enumext_renew_footnote_starred:
```

Now we insert our *sockets* for *tagging* PDF support and run \item.

```
4850            \__enumext_start_list_tag:n {enumext*}
4851            \__enumext_fake_make_label_vii:n {#1}
4852            \__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.

```
4853            \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
4854              \dim_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
4855              \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
4856              \__enumext_unskip_unkern:
4857              \__enumext_unskip_unkern:
4858              \skip_horizontal:n { -\l__enumext_listparindent_vii_dim } \ignorespaces
4859              \tl_use:N \l__enumext_fake_item_indent_vii_tl
4860              \tl_use:N \l__enumext_after_list_args_vii_tl
4861      }
```

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*.

```
4862 \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
4863   {
4864       \__enumext_endminipage:
4865     \__enumext_stop_list_tag:n {enumext*}
4866     \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*,.

```
4867       \int_set:Nn \hbadness { 10000 }
4868       \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`.

```
4869       \int_compare:nNnTF
4870         { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
4871         {
4872           \par\noindent
4873           \int_zero:N \l__enumext_item_column_pos_vii_int
4874         }
4875         {
4876           \skip_horizontal:N \l__enumext_columns_sep_vii_dim
4877         }
4878   }
```

(*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*⟩.

```
4879 \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
4880   {
4881     \int_compare:nNnT
4882       {
4883         \int_mod:nn
4884           {  \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
4885       }
4886       =
4887       { 0 }
4888       {
4889         \para_end:
4890         \skip_vertical:n { -\l__enumext_itemsep_vii_skip }
4891         \skip_vertical:N \c_zero_skip
4892         \int_gzero:N \g__enumext_item_count_all_vii_int
4893       }
4894   }
```

(*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`.

```
4895 \__enumext_after_env:nn {enumext*}
4896   {
4897       \__enumext_execute_after_env:
4898   }
```

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.

```
4899 \NewDocumentEnvironment{keyans*}{ o }
4900   {
4901     \__enumext_safe_exec_viii:
4902     \__enumext_parse_keys_viii:n {#1}
4903     \__enumext_before_list_viii:
4904     \__enumext_start_list:nn { }
4905       {
4906         \__enumext_list_arg_two_viii:
4907         \__enumext_before_keys_exec_viii:
4908       }
```

```
4909     \IfDocumentMetadataTF { \tag_suspend:n {keyans*} } { }
4910     \__enumext_starred_columns_set_viii:
4911     \item[] \scan_stop:
4912     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_first_item_tmp_viii:
4913     \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
4914     \ignorespaces
4915   }
4916   {
4917     \IfDocumentMetadataTF { \tag_struct_end:n {tag=text-unit} } { }
4918     \__enumext_stop_item_tmp_viii:
4919     \__enumext_remove_extra_parsep_viii:
4920     \__enumext_check_starred_cmd:n { item }
4921     \__enumext_after_list_viii:
4922   }
```

(*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.

```
4923 \cs_new_protected:Nn \__enumext_safe_exec_viii:
4924   {
4925     \bool_if:NF \l__enumext_store_active_bool
4926       {
4927         \msg_error:nnnn { enumext } { wrong-place }{ keyans* }{ save-ans }
4928       }
4929     \int_incr:N \l__enumext_keyans_level_h_int
4930     \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
4931       {
4932         \msg_error:nn { enumext } { nested }
4933       }
4934     \__enumext_keyans_name_and_start:
4935     \bool_if:NT \l__enumext_starred_bool
4936       {
4937         \msg_error:nnn { enumext } { nested-horizontal } { enumext* }
4938       }
4939     \bool_set_true:N \l__enumext_starred_bool
4940     % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
4941     \bool_set_false:N \l__enumext_store_active_bool
4942     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
4943       {
4944         \msg_error:nn { enumext } { keyans-wrong-level }
4945       }
4946   }
```

(*End of definition for* __enumext_safe_exec_viii:.)

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

```
4947 \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
4948   {
4949     \tl_if_novalue:nF {#1}
4950       {
4951         \keys_set:nn { enumext / keyans* } {#1}
4952       }
4953   }
```

(*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.

```
4954 \cs_new_protected:Nn \__enumext_before_list_viii:
4955   {
4956     \__enumext_vspace_above_viii:
4957     \__enumext_before_args_exec_viii:
4958     \__enumext_start_mini_viii:
4959   }
```

(*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.

```
4960 \cs_new_protected:Nn \__enumext_after_list_viii:
4961   {
4962     \__enumext_stop_mini_viii:
4963     \__enumext_after_stop_list_viii:
4964     \__enumext_vspace_below_viii:
4965   }
```

(*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.

```
4966 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_viii:
4967   {
4968     \skip_horizontal:n
4969       {
4970         -\l__enumext_labelwidth_viii_dim - \l__enumext_labelsep_viii_dim
4971       }
4972     \ignorespaces
4973   }
```

(*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`.

```
4974 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
4975   {
4976     \__enumext_stop_item_tmp_viii:
4977     \int_incr:N \l__enumext_item_column_pos_viii_int
4978     \int_gincr:N \g__enumext_item_count_all_viii_int
4979     \__enumext_item_peek_args_viii:
4980   }
```

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).

```
4981 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
4982   {
4983     \peek_meaning:NTF (
4984       { \__enumext_joined_item_viii:w }
4985       { \__enumext_joined_item_viii:w (1) }
4986   }
```

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`.

```
4987 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
4988   {
4989     \__enumext_starred_joined_item_viii:n {#1}
4990     \peek_meaning_remove:NTF *
4991       { \__enumext_starred_item_viii:w }
4992       { \__enumext_standar_item_viii:w }
4993   }
```

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 ]`.

```
4994  \cs_new_protected:Npn \__enumext_standar_item_viii:w
4995    {
4996      \bool_set_false:N \l__enumext_item_starred_viii_bool
4997      \bool_set_false:N \l__enumext_item_wrap_key_bool
4998      \peek_meaning:NTF [
4999        {
5000          \bool_set_eq:NN \l__enumext_wrap_label_viii_bool \l__enumext_wrap_label_opt_viii_bool
5001          \__enumext_start_item_viii:w
5002        }
5003        {
5004          \bool_set_true:N \l__enumext_wrap_label_viii_bool
5005          \legacy_if_set_true:n { @noitemarg }
5006          \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ] \ignorespaces
5007        }
5008    }
```

(*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⟩]`.

```
5009  \cs_new_protected:Npn \__enumext_starred_item_viii:w
5010    {
5011      \bool_set_true:N \l__enumext_item_starred_viii_bool
5012      \bool_set_true:N \l__enumext_item_wrap_key_bool
5013      \bool_set_true:N \l__enumext_wrap_label_viii_bool
5014      \peek_meaning:NTF [
5015        { \__enumext_starred_item_viii_aux_i:w }
5016        { \__enumext_starred_item_viii_aux_ii:w }
5017    }
```

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`.

```
5018  \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
5019    {
5020      \tl_clear:N \l__enumext_store_current_label_tl
5021      \tl_if_novalue:nF { #1 }
5022        {
5023          \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_viii_tl
5024            {
5025              \tl_put_right:NV \l__enumext_store_current_label_tl \l__enumext_store_keyans_item_opt_
5026              \tl_put_right:Nn \l__enumext_store_current_label_tl { #1 }
5027            }
5028          \tl_set:Nn \l__enumext_store_current_opt_arg_tl { #1 }
5029        }
5030      \__enumext_starred_item_viii_aux_ii:w
5031    }
5032  \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
5033    {
5034      \legacy_if_set_true:n { @noitemarg }
5035      \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ] \ignorespaces
5036    }
```

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.

```
5037  \cs_new_protected:Nn \__enumext_keyans_starred_item_star:
5038    {
5039      \tl_put_left:Ne \l__enumext_store_current_label_tl { \l__enumext_label_viii_tl }
5040      \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
5041      \__enumext_keyans_store_ref:
5042      \tl_put_left:Nn \l__enumext_store_current_label_tl { \item }
```

```
5043    \__enumext_keyans_addto_seq_link:
5044    \int_gincr:N \g__enumext_check_starred_cmd_int
5045    \dim_compare:nNnT { \l__enumext_mark_sym_sep_viii_dim } = { \c_zero_dim }
5046      {
5047        \dim_set:Nn \l__enumext_mark_sym_sep_viii_dim { \l__enumext_labelsep_viii_dim }
5048      }
5049    \bool_if:NT \l__enumext_show_answer_bool
5050      {
5051        \tl_set_eq:NN \l__enumext_mark_answer_sym_tl \l__enumext_mark_answer_sym_viii_tl
5052        \str_set_eq:NN \l__enumext_mark_position_str \l__enumext_mark_position_viii_str
5053        \__enumext_print_keyans_box:NN
5054          \l__enumext_labelwidth_viii_dim \l__enumext_mark_sym_sep_viii_dim
5055      }
5056    \bool_if:NT \l__enumext_show_position_bool
5057      {
5058        \tl_set:Ne \l__enumext_mark_answer_sym_tl
5059          {
5060            \group_begin:
5061              \exp_not:N \normalfont
5062              \exp_not:N \footnotesize [ \int_eval:n
5063                {
5064                  \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
5065                }
5066              ]
5067            \group_end:
5068          }
5069        \str_set_eq:NN \l__enumext_mark_position_str \l__enumext_mark_position_viii_str
5070        \__enumext_print_keyans_box:NN
5071          \l__enumext_labelwidth_viii_dim \l__enumext_mark_sym_sep_viii_dim
5072      }
5073  }
```

(*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.

```
5074  \cs_new_protected:Npn \__enumext_keyans_wraper_label_viii:n #1
5075    {
5076      \bool_lazy_all:nT
5077        {
5078          { \bool_if_p:N \l__enumext_wrap_label_viii_bool            }
5079          { \bool_if_p:N \l__enumext_show_answer_bool               }
5080          { \bool_if_p:N \l__enumext_item_wrap_key_bool             }
5081          { \cs_if_exist_p:N \__enumext_keyans_wrapper_item_viii:n }
5082        }
5083        {
5084          \cs_set_eq:NN
5085            \__enumext_wrapper_label_viii:n \__enumext_keyans_wrapper_item_viii:n
5086        }
5087      \bool_if:NTF \l__enumext_wrap_label_viii_bool
5088        {
5089          \__enumext_wrapper_label_viii:n {#1}
5090        }
5091        { #1 }
5092    }
5093  \cs_new_protected_nopar:Npn \__enumext_fake_make_label_viii:n #1
5094    {
5095      \legacy_if:nT { @noitemarg }
5096        {
5097          \legacy_if_set_false:n { @noitemarg }
5098          \legacy_if:nT { @nmbrlist }
5099            {
5100              \refstepcounter{enumXviii}
5101            }
5102        }
5103      \bool_if:NT \l__enumext_item_starred_viii_bool
5104        {
5105          \__enumext_keyans_starred_item_star:
5106        }
5107      \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
5108        {
5109          \tl_use:N \l__enumext_label_font_style_viii_tl
```

```
5110            \__enumext_keyans_wraper_label_viii:n {#1}
5111        }
5112      \skip_horizontal:N \l__enumext_labelsep_viii_dim \ignorespaces
5113    }
```

(*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*

__enumext_start_item_viii:w
__enumext_stop_item_viii:

The implementation at this is very similar to that of the enumext* environment.

```
5114 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
5115    {
5116      \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
5117      \hbox_set_to_wd:Nnw \l__enumext_item_text_viii_box
5118        {
5119          \l__enumext_joined_width_viii_dim
5120          + \l__enumext_labelwidth_viii_dim
5121          + \l__enumext_labelsep_viii_dim
5122        }
5123      \__enumext_renew_footnote_starred:
5124      \__enumext_start_list_tag:n {keyans*}
5125      \__enumext_fake_make_label_viii:n {#1}
5126      \__enumext_stop_start_list_tag:
5127      \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim  }
5128        \dim_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
5129        \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
5130        \__enumext_unskip_unkern:
5131        \__enumext_unskip_unkern:
5132        \skip_horizontal:n { -\l__enumext_listparindent_viii_dim } \ignorespaces
5133        \tl_use:N \l__enumext_fake_item_indent_viii_tl
5134        \bool_if:NT \l__enumext_item_starred_viii_bool
5135          {
5136            \__enumext_keyans_show_item_opt_viii:
5137          }
5138        \tl_use:N \l__enumext_after_list_args_viii_tl
5139    }
5140 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
5141    {
5142        \__enumext_endminipage:
5143      \__enumext_stop_list_tag:n {keyans*}
5144      \hbox_set_end:
5145      \int_set:Nn \hbadness { 10000 }
5146      \box_use_drop:N \l__enumext_item_text_viii_box
5147      \int_compare:nNnTF
5148        { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
5149        {
5150          \par\noindent
5151          \int_zero:N \l__enumext_item_column_pos_viii_int
5152        }
5153        {
5154          \skip_horizontal:N \l__enumext_columns_sep_viii_dim
5155        }
5156    }
```

(*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.

```
5157 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
5158    {
5159      \int_compare:nNnT
5160        {
5161          \int_mod:nn
5162            { \g__enumext_item_count_all_viii_int }
5163            { \l__enumext_columns_viii_int }
5164        }
5165        =
5166        { 0 }
5167        {
5168          \para_end:
5169          \skip_vertical:n { -\l__enumext_itemsep_viii_skip }
5170          \skip_vertical:N \c_zero_skip
5171          \int_gzero:N \g__enumext_item_count_all_viii_int
```

```
5172        }
5173    }
```

(*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.

```
5174  \NewDocumentCommand \getkeyans { m }
5175    {
5176      \exp_args:Ne \__enumext_getkeyans_aux:n
5177        { \tl_to_str:e { \text_expand:n {#1} } }
5178    }
```

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

```
5179  \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
5180    {
5181      \str_if_in:nnTF {#1} { : }
5182        {
5183          \use:e
5184            {
5185              \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
5186                { {##1} {##2} }
5187            }
5188          \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
5189        }
5190        { \msg_error:nnn { enumext } { missing-colon } {#1} }
5191    }
```

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*.

```
5192  \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
5193    {
5194      \prop_if_exist:cTF { g__enumext_#1_prop }
5195        {
5196          \prop_item:cn { g__enumext_#1_prop }{#2}
5197        }
5198        {
5199          \msg_error:nnn { enumext } { undefined-storage-anskey } {#1}
5200        }
5201    }
```

(*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*⟩].

```
5202  \keys_define:nn  { enumext / print }
5203    {
5204      print*  .code:n     = \keys_precompile:neN { enumext / enumext* }
5205                             { \__enumext_filter_save_key:n {#1} }
5206                             \l__enumext_print_keyans_starred_tl, % starred cmd
5207      print*  .initial:n = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5208                             rightmargin=0pt, listparindent=0pt, nosep, label=\arabic*.,
5209                             columns=2, first=\small, font=\small },
5210      print-1 .code:n     = \keys_precompile:neN { enumext / level-1 }
5211                             { \__enumext_filter_save_key:n {#1} }
5212                             \l__enumext_print_keyans_i_tl,
5213      print-1 .initial:n = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
```

```
5214                              rightmargin=0pt, listparindent=0pt, nosep, label=\arabic*.,
5215                              columns=2, first=\small, font=\small },
5216     print-2 .code:n      = \keys_precompile:neN { enumext / level-2 }
5217                              { \__enumext_filter_save_key:n {#1} }
5218                              \l__enumext_print_keyans_ii_tl,
5219     print-2 .initial:n   = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5220                              rightmargin=0pt, listparindent=0pt,  nosep, label=(\alph*),
5221                              first=\small, font=\small },
5222     print-3 .code:n      = \keys_precompile:neN { enumext / level-3 }
5223                              { \__enumext_filter_save_key:n {#1} }
5224                              \l__enumext_print_keyans_iii_tl,
5225     print-3 .initial:n   = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5226                              rightmargin=0pt, listparindent=0pt, nosep, label=\roman*.,
5227                              first=\small, font=\small },
5228     print-4 .code:n      = \keys_precompile:neN { enumext / level-4 }
5229                              { \__enumext_filter_save_key:n {#1} }
5230                              \l__enumext_print_keyans_iv_tl,
5231     print-4 .initial:n   = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5232                              rightmargin=0pt, listparindent=0pt, nosep, label=\Alph*.,
5233                              first=\small, font=\small },
5234     print-* .code:n      = \keys_precompile:neN { enumext / enumext* }
5235                              { \__enumext_filter_save_key:n {#1} }
5236                              \l__enumext_print_keyans_vii_tl, % starred nested
5237     print-* .initial:n   = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5238                              rightmargin=0pt, listparindent=0pt, nosep, label=\arabic*.,
5239                              first=\small, font=\small },
5240   }
```

🎯 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.

```
5241   \NewDocumentCommand \printkeyans { s O{} m }
5242     {
5243       \group_begin:
5244         \tl_use:N \l__enumext_print_keyans_i_tl
5245         \tl_use:N \l__enumext_print_keyans_ii_tl
5246         \tl_use:N \l__enumext_print_keyans_iii_tl
5247         \tl_use:N \l__enumext_print_keyans_iv_tl
5248         \tl_use:N \l__enumext_print_keyans_vii_tl
5249         \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
5250       \group_end:
5251     }
```

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.

```
5252   \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
5253     {
5254       \seq_if_exist:cTF { g__enumext_#3_seq }
5255         {
5256           \seq_if_empty:cF { g__enumext_#3_seq }
5257             {
```

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.

```
5258               \bool_if:nTF {#1}
5259                 {
5260                   \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
5261                     {
5262                       \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
5263                     }
5264                     {
5265                       \tl_use:N \l__enumext_print_keyans_starred_tl
5266                       \bool_set_true:N \l__enumext_base_line_fix_bool
```

```
5267                    \bool_set_true:N \l__enumext_print_keyans_star_bool
5268                    \begin{enumext*}[#2]
5269                      \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5270                    \end{enumext*}
5271                    \bool_set_false:N \l__enumext_base_line_fix_bool
5272                    \bool_set_false:N \l__enumext_print_keyans_star_bool
5273                  }
5274                }
```

Otherwise it will open the environment enumext passing the *optional argument* to the *"first level"* then map the *sequence.*

```
5275                {
5276                  \begin{enumext}[#2]
5277                    \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5278                  \end{enumext}
5279                }
5280              }
5281            }
5282          {
5283            \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
5284          }
5285      }
```

(*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.

```
5286  \cs_new:Npn \__enumext_filter_first_level:n #1
5287    {
5288      \use:e
5289        {
5290          \keyval_parse:NNn
5291            \__enumext_filter_first_level_key:n
5292            \__enumext_filter_first_level_pair:nn {#1}
5293        }
5294    }
```

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*.

```
5295  \cs_new:Npn \__enumext_filter_first_level_key:n #1
5296    {
5297      \str_case:nnF {#1}
5298        {
5299          { resume   } {}
5300          { resume*  } {}
5301        }
5302        { , { \exp_not:n {#1} } }
5303    }
```

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.

```
5304  \cs_new:Npn \__enumext_filter_first_level_pair:nn #1#2
5305    {
5306      \str_case:nnF {#1}
5307        {
5308          { series } {}
5309          { resume } {}
5310          { save-ans } {}
5311        }
5312        { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
5313    }
```

(*End of definition for* __enumext_filter_first_level:n, __enumext_filter_first_level_key:n, *and* __enumext_filter_first_level_pair:nn.)

Now define a *"meta families"* of ⟨keys⟩ to access from \setenumext.

```
5314  \keys_define:nn { enumext / meta-families }
```

```
5315        {
5316          enumext-1 .code:n =
5317                    {
5318                      \keys_set:ne { enumext / level-1 }
5319                        {
5320                          \__enumext_filter_first_level:n {#1}
5321                        }
5322                    } ,
5323          enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
5324          enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
5325          enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
5326          keyans    .code:n = { \keys_set:nn { enumext / keyans  } {#1} } ,
5327          enumext*  .code:n =
5328                    {
5329                      \keys_set:ne { enumext / enumext* }
5330                        {
5331                          \__enumext_filter_first_level:n {#1}
5332                        }
5333                    } ,
5334          keyans*   .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
5335          print*    .code:n = { \keys_set:nn { enumext / print   } { print*  = {#1} } } ,
5336          print-1   .code:n = { \keys_set:nn { enumext / print   } { print-1 = {#1} } } ,
5337          print-2   .code:n = { \keys_set:nn { enumext / print   } { print-2 = {#1} } } ,
5338          print-3   .code:n = { \keys_set:nn { enumext / print   } { print-3 = {#1} } } ,
5339          print-4   .code:n = { \keys_set:nn { enumext / print   } { print-4 = {#1} } } ,
5340          print-*   .code:n = { \keys_set:nn { enumext / print   } { print-* = {#1} } } ,
5341          unknown   .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
5342        }
```

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

```
5343    \seq_const_from_clist:Nn \c__enumext_all_families_seq
5344      {
5345        enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
5346        keyans*, print-1, print-2, print-3, print-4, print-*, print*,
5347      }
```

Now we define the user command \setenumext.

```
5348    \NewDocumentCommand \setenumext { O{enumext,1} +m }
5349      {
5350        \seq_clear:N \l__enumext_setkey_tmpa_seq
5351        \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
5352        \int_set:Nn \l__enumext_setkey_tmpa_int
5353          {
5354            \seq_count:N \l__enumext_setkey_tmpb_seq
5355          }
5356        \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
5357          {
5358            \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
5359            \seq_map_function:NN \l__enumext_setkey_tmpb_seq \__enumext_set_parse:n
5360            \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
5361              {
5362                \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
5363              }
5364          }
5365          {
5366            \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
5367          }
5368        \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
5369          { \seq_map_inline:Nn \c__enumext_all_families_seq }
5370          { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
5371          {
5372            \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
5373          }
5374      }
```

Internal functions used by the \setenumext command.

```
5375    \cs_new_protected:Npn \__enumext_set_parse:n #1
5376      {
5377        \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
5378        \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
5379          { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
5380        \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
```

The margin labels (near line 5348):
\setenumext
__enumext_set_parse:n
__enumext_set_error:nn

```
5381        {
5382           \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
5383             { \tl_trim_spaces:n {#1} }
5384        }
5385        { \__enumext_set_error:nn {#1} { } }
5386    }
5387 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
5388    { \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).

\setenumextmeta   First we will create a prop list \c__enumext_meta_paths_prop to handle the *optional argument*.
\c__enumext_meta_paths_prop
__enumext_add_meta_key:nnn
__enumext_def_meta_key:nnn
__enumext_def_meta_key:Vnn

```
5389 \prop_const_from_keyval:Nn \c__enumext_meta_paths_prop
5390    {
5391      {enumext,1} = level-1,
5392      {enumext,2} = level-2,
5393      {enumext,3} = level-3,
5394      {enumext,4} = level-4,
5395      {enumext*}  = enumext*
5396    }
```

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

```
5397 \NewDocumentCommand \setenumextmeta { s O{enumext,1} m +m }
5398    {
5399      \str_if_eq:eeTF { \tl_trim_spaces:n {#3} } { unknown }
5400        { \msg_error:nn { enumext } { prohibited-unknown } }
5401        {
5402          \bool_if:nTF {#1}
5403            {
5404              \int_step_inline:nn { 4 }
5405                { \__enumext_add_meta_key:nnn { enumext, ##1 } {#3} {#4} }
5406              \__enumext_add_meta_key:nnn { enumext* } {#3} {#4}
5407            }
5408            { \__enumext_add_meta_key:nnn {#2} {#3} {#4} }
5409        }
5410    }
```

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

```
5411 \cs_new_protected:Npn \__enumext_add_meta_key:nnn #1
5412    {
5413      \tl_set:Nn \l__enumext_meta_path_tl {#1}
5414      \tl_replace_all:Nnn \l__enumext_meta_path_tl {~} {}
5415      \prop_get:NVNTF
5416        \c__enumext_meta_paths_prop \l__enumext_meta_path_tl \l__enumext_meta_path_tl
5417        { \__enumext_def_meta_key:Vnn \l__enumext_meta_path_tl }
5418        {
5419          \msg_error:nnn { enumext } { unknown-set } {#1}
5420          \use_none:nn
5421        }
5422    }
5423 \cs_new_protected:Npn \__enumext_def_meta_key:nnn #1#2#3
5424    {
5425      \bool_lazy_or:nnTF
5426        { \keys_if_exist_p:nn { enumext / #1 } {#2} }
5427        { \keys_if_exist_p:nn { enumext / enumext* } {#2} }
5428        { \msg_error:nnn { enumext } { already-defined } {#2} }
5429        {
5430          \keys_define:nn { enumext / #1 }
5431            {
5432              #2 .meta:n = {#3},
5433              #2 .value_forbidden:n = true
5434            }
5435        }
5436    }
5437 \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.

\foreachkeyans

__enumext_parse_foreach_keys:nn
__enumext_parse_foreach_keys:n
__enumext_foreach_keyans:nn
__enumext_foreach_add_body:n

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

```
5438  \keys_define:nn { enumext / foreach }
5439    {
5440      before  .tl_set:N  = \l__enumext_foreach_before_tl,
5441      before  .value_required:n = true,
5442      after   .tl_set:N  = \l__enumext_foreach_after_tl,
5443      after   .value_required:n = true,
5444      start   .int_set:N = \l__enumext_foreach_start_int,
5445      start   .value_required:n = true,
5446      stop    .int_set:N = \l__enumext_foreach_stop_int,
5447      stop    .value_required:n = true,
5448      step    .int_set:N = \l__enumext_foreach_step_int,
5449      step    .value_required:n = true,
5450      wrapper .cs_set_protected:Np = \__enumext_foreach_wrapper:n #1,
5451      wrapper .value_required:n = true,
5452      sep     .tl_set:N  = \l__enumext_foreach_sep_tl,
5453      sep     .value_required:n = true,
5454      unknown .code:n    = { \__enumext_parse_foreach_keys:n {#1} }
5455    }
5456  \keys_precompile:nnN { enumext / foreach }
5457    {
5458      before={},after={},start=1,step=1,stop=0,wrapper=#1,sep={; }
5459    }
5460    \l__enumext_foreach_default_keys_tl
```

Functions for handling unknown ⟨*keys*⟩.

```
5461  \cs_new_protected:Npn \__enumext_parse_foreach_keys:nn #1#2
5462    {
5463      \tl_if_blank:nTF {#2}
5464        {
5465          \msg_error:nnn { enumext } { for-key-unknown } {#1}
5466        }
5467        {
5468          \msg_error:nnnn { enumext } { for-key-value-unknown } {#1} {#2}
5469        }
5470    }
5471  \cs_new_protected:Npn \__enumext_parse_foreach_keys:n #1
5472    {
5473      \exp_args:NV \__enumext_parse_foreach_keys:nn \l_keys_key_str {#1}
5474    }
```

We create the command.

```
5475  \NewDocumentCommand \foreachkeyans { +O{} m }
5476    {
5477      \__enumext_foreach_keyans:nn {#1} {#2}
5478    }
```

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

```
5479  \cs_new_protected:Npn \__enumext_foreach_keyans:nn #1 #2
5480    {
5481      \tl_use:N \l__enumext_foreach_default_keys_tl
5482      \keys_set:nn { enumext / foreach } {#1}
5483      \tl_set:Nn \l__enumext_foreach_name_prop_tl {#2}
5484      \prop_if_exist:cF { g__enumext_#2_prop }
5485        {
5486          \msg_error:nnn { enumext } { undefined-storage-anskey } {#2}
5487        }
5488      \int_compare:nNnT { \l__enumext_foreach_stop_int } = { 0 }
5489        {
5490          \int_set:Nn \l__enumext_foreach_stop_int
5491            { \prop_count:c { g__enumext_#2_prop } }
5492        }
5493      \seq_clear:N \l__enumext_foreach_print_seq
```

```
5494        \int_step_function:nnnN
5495          { \l__enumext_foreach_start_int }
5496          { \l__enumext_foreach_step_int }
5497          { \l__enumext_foreach_stop_int }
5498          \__enumext_foreach_add_body:n
5499          \seq_use:NV \l__enumext_foreach_print_seq \l__enumext_foreach_sep_tl
5500        }
5501 \cs_new_protected:Npn \__enumext_foreach_add_body:n #1
5502      {
5503        \seq_put_right:Ne \l__enumext_foreach_print_seq
5504          {
5505            \exp_not:V \l__enumext_foreach_before_tl
5506            \__enumext_foreach_wrapper:n
5507              {
5508                \prop_item:cn { g__enumext_ \l__enumext_foreach_name_prop_tl _prop }{#1}
5509              }
5510            \exp_not:V \l__enumext_foreach_after_tl
5511          }
5512      }
```

(*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.

```
5513 \msg_new:nnn { enumext } { package-load }
5514    {
5515      The~'#1'~package~is~already~loaded.
5516    }
5517 \msg_new:nnn { enumext } { package-not-load }
5518    {
5519      The~'#1'~package~will~be~loaded~as~a~dependency.
5520    }
5521 \msg_new:nnn { enumext } { package-load-foot }
5522    {
5523      The~'#1'~package~is~loaded~with~the~option~'#2'.
5524    }
```

Message used in the creation of counters by enumext package.

```
5525 \msg_new:nnn { enumext } { counters }
5526    {
5527      The~counter~'#1'~is~already~defined~by~some~\\
5528      package~or~macro,~it~cannot~be~continued.
5529    }
```

Message used by align and mark-pos keys.

```
5530 \msg_new:nnn { enumext } { unknown-choice }
5531    {
5532      The~value~'#3'~for~'#1'~key~is~invalid~use~('#2').
5533    }
```

Message used by reserved anskey* environment by enumext package.

```
5534 \msg_new:nnnn { enumext } { anskey-env-error }
5535    {
5536      The~environment~'#1'~is~reserved~by ~\\
5537      'enumext'~package,~It~is~already~defined.
5538    }
5539    {
5540      The~environment~'#1'~is~defined~internally ~
5541      for~the~'save-ans'~key~with~save-ans~key~active.~See~documentation.\\
5542    }
5543 \msg_new:nnn { enumext } { anskey-env-nested }
5544    {
5545      The~#1~'#2'~can't~be~nested~\msg_line_context:.
5546    }
```

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

```
5547 \msg_new:nnn { enumext } { store-prop }
5548    {
5549      *~Package~enumext:~Creating ~
5550        \c_backslash_str g__enumext_#1_prop~\msg_line_context:.
5551    }
5552 \msg_new:nnn { enumext } { store-seq }
```

```
5553      {
5554        *~Package~enumext:~Creating ~
5555        \c_backslash_str g__enumext_#1_seq~\msg_line_context:.
5556      }
5557  \msg_new:nnn { enumext } { store-int }
5558      {
5559        *~Package~enumext:~Creating ~
5560        \c_backslash_str g__enumext_resume_#1_int~\msg_line_context:.
5561      }
5562  \msg_new:nnn { enumext } { prop-seq-int-hook }
5563      {
5564        *~Package~enumext:~Elements~in ~
5565        \c_backslash_str g__enumext_#1_prop~=~#2.\\
5566        *~Package~enumext:~Elements~in ~
5567        \c_backslash_str g__enumext_#1_seq~=~#3.\\
5568        *~Package~enumext:~Value~off ~
5569        \c_backslash_str g__enumext_resume_#1_int~=~#4.
5570      }
5571  \msg_new:nnn { enumext } { item-answer-hook }
5572      {
5573        *~Package~enumext:~Value~off ~
5574        \c_backslash_str g__enumext_item_number_int~=~#1.\\
5575        *~Package~enumext:~Value~off ~
5576        \c_backslash_str g__enumext_item_anskey_int~=~#2.\\
5577        *~Package~enumext:~Difference~item_number_int~-~item_anskey_int~=~#3.
5578      }
```

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

```
5579  \msg_new:nnn { enumext } { invalid-key }
5580      {
5581        The~key~'#1'~is~not~know~the~level~#2.
5582      }
5583  \msg_new:nnn { enumext } { unknown-key-family }
5584      {
5585        Unknown~key~family~`\l_keys_key_str'~for~enumext.
5586      }
```

Messages used in length calculation.

```
5587  \msg_new:nnn { enumext } { width-negative }
5588      {
5589        Ignoring~negative~value~'#1=#2'~\msg_line_context:.\\
5590        The~key~'#1'~ accepts~values ~>=~0pt.
5591      }
5592  \msg_new:nnn { enumext } { width-zero }
5593      {
5594        Invalid~'#1=#2'~\msg_line_context:.\\
5595        The~key~'#1'~ accepts~values ~>~0pt.
5596      }
```

Messages used by show-length key in enumext.

```
5597  \msg_new:nnn { enumext } { list-lengths }
5598      {
5599        ****~Lengths~used~by~'enumext'~level~'#2'~\msg_line_context:~\c_space_tl ****\\
5600        \__enumext_show_length:nnn { dim  } { labelsep     } {#1}
5601        \__enumext_show_length:nnn { dim  } { labelwidth   } {#1}
5602        \__enumext_show_length:nnn { dim  } { itemindent   } {#1}
5603        \__enumext_show_length:nnn { dim  } { leftmargin   } {#1}
5604        \__enumext_show_length:nnn { dim  } { rightmargin  } {#1}
5605        \__enumext_show_length:nnn { dim  } { listparindent } {#1}
5606        \__enumext_show_length:nnn { skip } { topsep     } {#1}
5607        \__enumext_show_length:nnn { skip } { parsep     } {#1}
5608        \__enumext_show_length:nnn { skip } { partopsep } {#1}
5609        \__enumext_show_length:nnn { skip } { itemsep   } {#1}
5610        ************************************************
5611      }
```

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

```
5612  \msg_new:nnn { enumext } { list-lengths-not-nested }
5613      {
5614        ****~Lengths~used~by~'#2'~environment~\msg_line_context:~\c_space_tl ****\\
5615        \__enumext_show_length:nnn { dim  } { labelsep     } {#1}
5616        \__enumext_show_length:nnn { dim  } { labelwidth   } {#1}
5617        \__enumext_show_length:nnn { dim  } { itemindent   } {#1}
```

```
5618    \__enumext_show_length:nnn { dim  } { leftmargin   } {#1}
5619    \__enumext_show_length:nnn { dim  } { rightmargin  } {#1}
5620    \__enumext_show_length:nnn { dim  } { listparindent } {#1}
5621    \__enumext_show_length:nnn { skip } { topsep     } {#1}
5622    \__enumext_show_length:nnn { skip } { parsep     } {#1}
5623    \__enumext_show_length:nnn { skip } { partopsep } {#1}
5624    \__enumext_show_length:nnn { skip } { itemsep    } {#1}
5625    *************************************************
5626    }
```

Messages used by `ref` key.

```
5627 \msg_new:nnn { enumext } { key-ref-empty }
5628   {
5629     Key~'ref'~need~a~value~in~'#1'~ \msg_line_context:.
5630   }
```

Messages used by `save-ans` key.

```
5631 \msg_new:nnn { enumext } { save-ans-empty }
5632   {
5633     Key~'save-ans'~need~a~value~in~'#1'~ \msg_line_context:.
5634   }
5635 \msg_new:nnn { enumext } { save-ans-log }
5636   {
5637     *~Package~enumext:~Start~#1\c_space_tl with~save-ans=#2~\msg_line_context:.
5638   }
5639 \msg_new:nnn { enumext } { save-ans-log-hook }
5640   {
5641     *~Package~enumext:~Stop~#1\c_space_tl with~save-ans=#2~\msg_line_context:.
5642   }
5643 \msg_new:nnn { enumext } { save-ans-hook }
5644   {
5645     Stop~storing~for~'save-ans=#1'~\msg_line_context:.
5646   }
```

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

```
5647 \msg_new:nnn { enumext } { need-save-ans }
5648   {
5649     Key~'#1'~ works~only~with~the~'save-ans'~key~in~'#2'~ \msg_line_context:.
5650   }
5651 \msg_new:nnn { enumext } { items-same-answer }
5652   {
5653     *****************************************\\
5654     *~Package~enumext:~Checking~answers~in~'#1' ~
5655     for~\c_left_brace_str #2 \c_right_brace_str\\
5656     *~started~#3~and~close~\msg_line_context: : ~
5657     'OK',~all~items~with~answer.\\
5658     *****************************************
5659   }
5660 \msg_new:nnn { enumext } { item-greater-answer }
5661   {
5662     Checking~answers~in~'#1'~for~\c_left_brace_str #2 \c_right_brace_str\\
5663     started~#3~and~close~\msg_line_context: : ~'NOT~OK'\\
5664     Items~>~Answers.
5665   }
5666 \msg_new:nnn { enumext } { item-less-answer }
5667   {
5668     Checking~answers~in~'#1'~for~\c_left_brace_str #2 \c_right_brace_str\\
5669     started~#3~and~close~\msg_line_context: : ~'NOT~OK'\\
5670     Items~<~Answers.
5671   }
```

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

```
5672 \msg_new:nnn { enumext } { missing-starred }
5673   {
5674     Missing~'\c_backslash_str #1*'~#2.
5675   }
5676 \msg_new:nnn { enumext } { many-starred }
5677   {
5678     Many~'\c_backslash_str #1*'~#2.
5679   }
```

Messages used by `\printkeyans*` command.

```
5680 \msg_new:nnn { enumext } { print-starred }
```

```
5681    {
5682        \c_backslash_str printkeyans*:~ The~sequence~'#1'~already~contains ~
5683        #2~environment~ \msg_line_context:.
5684    }
```

Message for the nesting depth of the environment enumext.

```
5685 \msg_new:nnn { enumext } { list-too-deep }
5686    {
5687        Too~deep~nesting ~for~'enumext'~\msg_line_context:.~ \\
5688        The~maximum ~level ~of ~nesting ~is~4.
5689    }
```

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

```
5690 \msg_new:nnn { enumext } { anskey-unnumber-item }
5691    {
5692        Can't~store~with~a~unnumbered~\c_backslash_str item~\msg_line_context:.
5693    }
5694 \msg_new:nnn { enumext } { anskey-already-stored }
5695    {
5696        Content~already~stored~for~this~\c_backslash_str item~\msg_line_context:.
5697    }
5698 \msg_new:nnn { enumext } { anskey-empty-arg }
5699    {
5700        Can't~store~empty~content~\msg_line_context:.
5701    }
5702 \msg_new:nnn { enumext } { anskey-wrong-place }
5703    {
5704        Wrong~place~for~command~'\c_backslash_str #1'~\msg_line_context:.~ \\
5705        '\c_backslash_str #1'~works~in~the~environment~'#2'.
5706    }
5707 \msg_new:nnn { enumext } { anskey-nested }
5708    {
5709        The~command~\c_backslash_str anskey~ can't~be~nested~\msg_line_context:.
5710    }
5711 \msg_new:nnn { enumext } { anskey-math-mode }
5712    {
5713        #1~can't~work~in~math~mode~\msg_line_context:.
5714    }
5715 \msg_new:nnn { enumext } { anskey-env-wrong }
5716    {
5717        The~environment~anskey*~cannot~use~in~'#1'~\msg_line_context:.
5718    }
5719 \msg_new:nnn { enumext } { anspic-wrong-place }
5720    {
5721        Wrong~place~for~command~'\c_backslash_str #1'~\msg_line_context:.~ \\
5722        '\c_backslash_str #1'~works~in~the~environment~'#2'.
5723    }
5724 \msg_new:nnn { enumext } { command-wrong-place }
5725    {
5726        Wrong~place~for~command~'\c_backslash_str #1'~\msg_line_context:.~ \\
5727        '\c_backslash_str #1'~works~outside~the~environment~'#2'.
5728    }
5729 \msg_new:nnnn { enumext } { anskey-env-key-unknown }
5730    {
5731      The~key~'#1'~is~unknown~by~environment~
5732      'anskey*'~and~is~being~ignored.
5733    }
5734    {
5735      The~environment~'anskey*'~does~not~have~a~key~called ~'#1'.\\
5736      Check~that~you~have~spelled~the~key~name~correctly.
5737    }
5738 \msg_new:nnnn { enumext } { anskey-env-key-value-unknown }
5739    {
5740      The~key~'#1=#2'~is~unknown~by~environment ~
5741      'anskey*'~and~is~being~ignored.
5742    }
5743    {
5744      The~environment~'anskey*'~does~not~have~a~key~called ~'#1'.\\
5745      Check~that~you~have~spelled~the~key~name~correctly.
5746    }
5747 \msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
5748    { The~key~'#1'~is~unknown~by~'\c_backslash_str anskey'~and~is~being~ignored.}
```

```
5749    {
5750      The~command ~'\c_backslash_str anskey'~does~not~have~a~key~called ~'#1'.\\
5751      Check~that~you~have~spelled~the~key~name~correctly.
5752    }
5753  \msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
5754    { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str anskey'~and~is~being~ignored. }
5755    {
5756      The~command~'\c_backslash_str anskey'~does~not~have~a~key~called ~'#1'.\\
5757      Check~that~you~have~spelled~the~key~name~correctly.
5758    }
5759  \msg_new:nnn { enumext } { overwrite-file }
5760    {
5761      Overwriting~file~'#1'.
5762    }
5763  \msg_new:nnn { enumext } { writing-file }
5764    {
5765      Writing~file~'#1'.
5766    }
5767  \msg_new:nnn { enumext } { not-writing }
5768    {
5769      File~`#1'~already~exists.~Not~writing.
5770    }
```

Messages used by keyans, keyans* and keyanspic environment.

```
5771  \msg_new:nnn { enumext } { keyans-nested }
5772    {
5773      The~environment~'keyans'~can't~be ~nested ~\msg_line_context:.
5774    }
5775  \msg_new:nnn { enumext } { keyans-wrong-level }
5776    {
5777      Wrong~level~position~for~'keyans'~\msg_line_context:.~ \\
5778      The~environment~'keyans'~can~only~be~in~the~first~level.
5779    }
5780  \msg_new:nnn { enumext } { wrong-place }
5781    {
5782      Wrong~place~for~'#1'~environment ~\msg_line_context:.~ \\
5783      '#1'~is~only~found~with~'#2'~ in ~ 'enumext.
5784    }
5785  \msg_new:nnn { enumext } { keyanspic-nested }
5786    {
5787      The~environment~'keyanspic'~can't~be ~nested~ \msg_line_context:.~.
5788    }
5789  \msg_new:nnn { enumext } { keyanspic-wrong-level }
5790    {
5791      Wrong~level~position~for~'keyanspic'~\msg_line_context:.~ \\
5792      The~environment~'keyans'~can~only~be~in~the~first~level.
5793    }
5794  \msg_new:nnn { enumext } { keyanspic-item-cmd }
5795    {
5796      Can't~use ~\c_backslash_str item~in~keyanspic~\msg_line_context:.
5797    }
5798  \msg_new:nnnn { enumext } { keyans-unknown-key }
5799    {
5800      The~key~'#1'~is~unknown~by~environment~
5801      '\l__enumext_envir_name_tl'~and~is~being~ignored.
5802    }
5803    {
5804      The~environment~'\l__enumext_envir_name_tl'~does~not
5805     ~have~a~key~called ~'#1'.\\
5806      Check~that~you~have~spelled~the~key~name~correctly.
5807    }
5808  \msg_new:nnnn { enumext } { keyans-unknown-key-value }
5809    {
5810      The~key~'#1=#2'~is~unknown~by~environment ~
5811      '\l__enumext_envir_name_tl'~and~is~being~ignored.
5812    }
5813    {
5814      The~environment~'\l__enumext_envir_name_tl'~does~not
5815     ~have~a~key~called ~'#1'.\\
5816      Check~that~you~have~spelled~the~key~name~correctly.
5817    }
```

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

```
5818 \msg_new:nnnn { enumext } { starred-unknown-key }
5819   {
5820     The~key~'#1'~is~unknown~by~environment~
5821     '\l__enumext_envir_name_tl'~and~is~being~ignored.
5822   }
5823   {
5824     The~environment~'\l__enumext_envir_name_tl'~does~not
5825   ~have~a~key~called ~'#1'.\\
5826     Check~that~you~have~spelled~the~key~name~correctly.
5827   }
5828 \msg_new:nnnn { enumext } { starred-unknown-key-value }
5829   {
5830     The~key~'#1=#2'~is~unknown~by~environment ~
5831     '\l__enumext_envir_name_tl'~and~is~being~ignored.
5832   }
5833   {
5834     The~environment~'\l__enumext_envir_name_tl'~does~not
5835   ~have~a~key~called ~'#1'.\\
5836     Check~that~you~have~spelled~the~key~name~correctly.
5837   }
```

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

```
5838 \msg_new:nnnn { enumext } { standar-unknown-key }
5839   {
5840     The~key~'#1'~is~unknown~by~environment~'\l__enumext_envir_name_tl' \c_space_tl
5841   ~on~level~\int_use:N \l__enumext_level_int \c_space_tl and~is~being~ignored.
5842   }
5843   {
5844     The~environment~'\l__enumext_envir_name_tl'~does~not
5845   ~have~a~key~called ~'#1'~on~level~\int_use:N \l__enumext_level_int.\\
5846     Check~that~you~have~spelled~the~key~name~correctly.
5847   }
5848 \msg_new:nnnn { enumext } { standar-unknown-key-value }
5849   {
5850     The~key~'#1=#2'~is~unknown~by~environment~'\l__enumext_envir_name_tl' \c_space_tl
5851   ~on~level~\int_use:N \l__enumext_level_int \c_space_tl and~is~being~ignored.
5852   }
5853   {
5854     The~environment~'\l__enumext_envir_name_tl'~does~not
5855   ~have~a~key~called ~'#1'~on~level~\int_use:N \l__enumext_level_int.\\
5856     Check~that~you~have~spelled~the~key~name~correctly.
5857   }
```

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

```
5858 \msg_new:nnnn { enumext } { for-key-unknown }
5859   { The~key~'#1'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored.}
5860   {
5861     The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5862     Check~that~you~have~spelled~the~key~name~correctly.
5863   }
5864 \msg_new:nnnn { enumext } { for-key-value-unknown }
5865   { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored. }
5866   {
5867     The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5868     Check~that~you~have~spelled~the~key~name~correctly.
5869   }
```

Messages used by \getkeyans command.

```
5870 \msg_new:nnn { enumext } { undefined-storage-anskey }
5871   {
5872     Storage~named~'#1'~is~not~defined~\msg_line_context:.
5873   }
```

Messages used by \miniright command.

```
5874 \msg_new:nnn { enumext } { missing-miniright }
5875   {
5876     Missing~'\c_backslash_str miniright'~in~\msg_line_context:.\\
5877     The~key~'mini-env'~need~'\c_backslash_str miniright'.
5878   }
5879 \msg_new:nnn { enumext } { wrong-miniright-place }
5880   {
5881     Wrong~place~for~'\c_backslash_str miniright'~\msg_line_context:.~ \\
```

```
5882        Works~in~'enumext'~and~'keyans'~with~key~'mini-env'.
5883    }
5884 \msg_new:nnn { enumext } { wrong-miniright-use }
5885    {
5886      Wrong~use~for~'\c_backslash_str miniright'~\msg_line_context:.~ \\
5887      '\c_backslash_str miniright'~need~a~key~'mini-env'.
5888    }
5889 \msg_new:nnn { enumext } { wrong-miniright-starred }
5890    {
5891      Can't~use ~\c_backslash_str miniright~in~starred~environments~\msg_line_context:.
5892    }
5893 \msg_new:nnn { enumext } { many-miniright-used }
5894    {
5895      Can't~use ~\c_backslash_str miniright~more~than~once~ \msg_line_context:.
5896    }
```

Messages used by \setenumextmeta command.

```
5897 \msg_new:nnn { enumext } { unknown-set }
5898    {
5899      Argument~[#1]~is~unknown~by~ \c_backslash_str setenumextmeta~\msg_line_context:.
5900    }
5901 \msg_new:nnn { enumext } { already-defined }
5902    {
5903      The~key~'#1'~is~already~defined~\msg_line_context:.
5904    }
5905 \msg_new:nnn { enumext } { prohibited-unknown }
5906    {
5907      The~name~'unknown'~can't~be~chosen~ for~a~meta~key~\msg_line_context:.
5908    }
```

Messages used by enumext* and keyans* environments.

```
5909 \msg_new:nnn { enumext } { nested }
5910    {
5911      The~environment~\l__enumext_envir_name_tl \c_space_tl can't~be~nested~\msg_line_context:.
5912    }
5913 \msg_new:nnn { enumext } { nested-horizontal }
5914    {
5915      The~environment~\l__enumext_envir_name_tl \c_space_tl can't~be~nested~in~'#1'~ \msg_line_cont
5916    }
5917 \msg_new:nnn { enumext } { item-joined }
5918    {
5919      Items~joined~(#1)~>~#2 ~columns ~\msg_line_context:.
5920    }
5921 \msg_new:nnn { enumext } { item-joined-columns }
5922    {
5923      Not~space~to~join~items~(#1)~>~#2 ~\msg_line_context:.
5924    }
```

13.52 Finish package

Finish package implementation.

```
5925 \file_input_stop:
5926 ⟨/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.