enumext
ENUMERATE EXERCISE SHEETS

V1.0   2024-09-29*

©2024 by Pablo González†

Abstract

This package provides "enumerated list" environments compatible with LaTeX tagging PDF for creating *"simple exercise sheets"* along with *"multiple choice questions"*, storing the ⟨*answers*⟩ to these in memory using `multicol` and `scontents` packages and the `l3seq` and `l3prop` modules.

Contents

Motivation and acknowledgments

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

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

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

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

---

*This file describes a documentation for v1.0, last revised 2024-09-29.
†E-mail: «pablgonz@educarchile.cl».

License and Requirements

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

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

1   Introduction

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

In my work (and in the work of many teachers) it is common to use *"simple exercise sheets"* also known as *"informal lists of exercises"*, as an example:

1. Factor $x^2 - 2x + 1$
2. Factor $3x + 3y + 3z$
3. True False
   (a) $\alpha > \delta$
   (b) LaTeX2e is cool?
4. Related to Linux

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

Sometimes we are also interested in showing the *"answers"* along with the questions:

1. Factor $x^2 - 2x + 1$
   * $(x-1)^2$
2. Factor $3x + 3y + 3z$
   * $3(x + y + z)$
3. True False
   (a) $\alpha > \delta$
      * False
   (b) LaTeX2e is cool?
      * Very True!
4. Related to Linux

(a) You use linux?
   * Yes
(b) Usually uses the package manager?
   * Yes, `dnf`
(c) Rate the following package and class
   i.   `xsim-exam`
      * doesn't exist for now :(
   ii. `xsim`
      * very good
   iii. `exsheets`
      * obsolete

Or we are interested in referring to a specific question and its *"answer"*, for example:

The answer to 3.(b) is "Very True!" and the answer to 4.(c).ii is "very good".

Or we are interested in printing all the *"answers"*:

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

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

Another very common thing to use in my work is *"multiple choice questions"*, for example:

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

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

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

4. Question with image and label below:
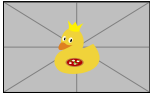

A)   B)   C)

D)   E)

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

Where what we are interested in the ⟨*label*⟩ and a *"short note"* that we leave as an explanation, and then print them:

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

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

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

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

1.1 Description and usage

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

🎯 This package is NOT intend to replace the `enumerate` environment nor replace the powerful `enumitem`[6], the approach is intended to work without hindering either of them.

This package can be used with `xelatex`, `lualatex`, `pdflatex` and the classical `latex»dvips»ps2pdf` and is present in TeX Live and MiKTeX, use the package manager to install. For manual installation, download enumext.zip and unzip it, run `lualatex enumext.dtx` and move all files to appropriate locations, then run `mktexlsr`. To produce the documentation run `lualatex enumext.dtx` two times.

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

The package is loaded in the usual way:

```
\usepackage{enumext}
```

1.2 The concept of left margin

There is a direct relationship between the parameters `\leftmargin`, `\itemindent`, `\labelwidth` and `\labelsep` plus an *"extra space"* that makes it difficult to obtain the desired *horizontal spaces* in a `list` environment.

Usually we don't want the `list` to go beyond the left margin of the page, but since these four values are related, that causes a problem. The `enumitem`[6] package adds the `\labelindent` parameter to solve some of these problems. A simplified representation of this in the figure 1.

Figure 1: Representation of horizontal lengths in `enumitem`.

The enumext package does NOT provide a user interface to set the values for `\leftmargin` and `\itemindent`, instead it provides the keys `list-offset` and `list-indent` which internally set the values for `\leftmargin` and `\itemindent`. The concepts of `\leftmargin` and `\itemindent` are different in enumext. The figure 2 shows the visual representation of idea.

Figure 2: Representation of horizontal lengths concept in enumext.

In this way we reduce a *little* the amount of parameters we have to pass. With the default values of keys `list-offset`, `list-indent`, `labelwidth` and `labelsep` the lists will have the (usually) expected output for *"simple worksheets"*. The figure 3 shows the visual representation.

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

1.3   User interface

The user interface consists of two main list environments enumext (vertical) and enumext* (horizontal), the environment anskey* and the command \anskey to *"store content"* and the environments keyans, keyans* and keyanspic for multiple choice. It also provides the commands \getkeyans to print individual *stored content*, \printkeyans to print all *stored content*, \miniright for minipage and \setenumext to config all [⟨*key* = *val*⟩] options.

1.3.1   Internal counters

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

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

1.3.2   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

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

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

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

🏷 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*.

enumext
enumext*

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

2.1   The environment `enumext`

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

Example with `columns=2`

1. This text is in the first level.

  (a) This text is in the second level.

    i. This text is in the third level.

A. This text is in the fourth level.

X This text is in the first level.

⋆ 2. This text is in the first level.

2.2   The environment `enumext*`

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

Some considerations to take into account for this environment:
- The environment cannot be nested within itself or in the environment keyans*, but it can be nested within enumext and vice versa.
- Each *"item"* in the environment is placed within a `minipage` environment whose *width* is stored in the dimension `\itemwidth` that NOT includes `labelwith`, `labelsep`, only the *width of the content*.
- You cannot have floating environments like `figure` or `table` but `\footnote` with hyperref support is supported if the `footnotehyper` package is loaded.
- 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*`

\item*

```
\item*
\item*[⟨symbol⟩]
\item*[⟨symbol⟩][⟨offset⟩]
```

The `\item*`, `\item*[⟨symbol⟩]` and `\item*[⟨symbol⟩][⟨offset⟩]` works like the numbered `\item`, but placing a ⟨symbol⟩ to the *"left"* of the ⟨label⟩ separated from it by the ⟨offset⟩ set by the the *second optional argument*. The default values for ⟨symbol⟩ and ⟨offset⟩ are `$\star$` '⋆' and the value set by `labelsep` key.

The *starred argument* '*' cannot be separated by spaces '␣' from the command, i.e. `\item*` and the *first optional argument* does "NOT" support *verbatim content*. Can be configure with the keys `item-sym` and `item-pos` locally in the environment or globally using `\setenumext` command (§3).

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

2.3.1 Keys for \item*

item-sym* = {⟨symbol⟩} default: *$\star$*

Sets the *symbol* to be displayed in the *"left"* of the box containing the current ⟨label⟩ set by labelwidth key for \item* in enumext and enumext*. The *symbol* can be in text or math mode, for example item-sym*={$\ast$}.

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

Sets the *offset* between the box containing the current ⟨label⟩ defined by labelwidth key and the ⟨symbol⟩ set by item-sym* key. The default values are set by labelsep key at each level. If positive values are passed it will *offset to the left* and if negative values are passed it will *offset to the right*.

2.4 The command \item in enumext*

The \item command for the enumext* environment provides an *"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 9. The ninth
two columns (196.17749pt)

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 \printkeyans command. It can be used both in the preamble and in the body of the document as many times as desired.

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

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

4 The command \setenumextmeta

\setenumextmeta

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

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

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

5 The `keyval` system

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

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

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

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

5.1 Keys for `label` and `ref`

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

Sets the ⟨*label*⟩ that will be printed at the *current level*. The default value for the first level of the environments enumext and enumext* are \arabic*., for second level are (\alph*), for third level are \roman*. and for fourth level are \Alph*.. For keyans and keyans* environments the default value is \Alph*).

🛈 This key is intended to give the basic structure with which the ⟨*label*⟩ will be displayed, and the form in which it is used by standard *"label and ref"* and the *"internal reference"* system with the save-ref key. You cannot use commands with ⟨*label*⟩ as an argument, for example \emph{⟨\alph*⟩} will return an error. For full customization of how ⟨*label*⟩ is displayed use the font or wrap-label keys.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Wraps the *current* ⟨*label*⟩ defined by label key referenced by {#1}. The {⟨*code*⟩} must be passed between braces. This key does not modify the value set by the labelwidth key and is applied only on \item and \item*. When using it in the \setenumext command it is necessary to use the *double hash* '{##1}'. For example wrap-label={\fbox{#1}} or you can create a command:

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

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

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

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

5.2 Keys for spaces

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

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

5.2.1 Vertical spaces

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

5.2.2   Horizontal spaces

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

Extra *horizontal indentation*, beyond `labelsep`, of the *"first line"* off each item. This value is applied internally using `\hspace` and does not modify the value of `\itemindent`.

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

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

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

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

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

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

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

Sets the *indentation* of the whole environment under the box defined by `labelwidth` and `labelsep` keys. Internally sets the value of `\leftmargin` and `\itemindent` for the current level.

If `list-indent=0pt` is set in the environment `enumext` the ⟨*label*⟩ will be part of the text, separated by the value of the `labelsep` key and the *first word*, in simple terms it will look like a *"common paragraph"*. This setting is equivalent (more or less) to the `wide` key provided by the `enumitem` package.

💣 For the `enumext*` and `keyans*` environments the keys `list-indent` and `list-offset` have the same effect.

5.3   Keys for add code

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

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

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

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

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

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

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

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

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

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

5.4   Keys for `start`, `series` and `resume`

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

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

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

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

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

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

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

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

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

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

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

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

5.5 Keys for `multicols`

columns = {⟨integer⟩}   default: `1`

Set the *number of columns* to be used by the `multicols` environment within the environment. The value must be a positive integer less than or equal to `10`.

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

Set the *space between* columns used by the `multicols` environment within the environment. Internally sets the value of `\columnsep`, by default its value is equal to the sum of the values set in the keys `labelwidth` and `labelsep` of the current level.

🔹 The `\footnote`{⟨text⟩} command in the nested levels of `multicols` will not work as expected, prefer the use of `\footnotemark`[⟨number⟩] inside the environment and `\footnotetext`[⟨number⟩]{⟨text⟩} outside the environment or via the `after` key.

5.6 Keys for `minipage`

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

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

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

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

5.6.1 The command `\miniright`

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

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

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

🔹 The `\footnote`{⟨text⟩} command in `minipage` environment will work as usual. If you prefer the footnotes to be numbered (not lowercase) and outside the environment, use `\footnotemark`[⟨number⟩] inside the environment and `\footnotetext`[⟨number⟩]{⟨text⟩} outside the environment or via the `after` key (see §1.3.6 for full support).

5.6.2 The key `mini-right`

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

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

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

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

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

6 The storage system

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

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

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

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

6.1 Keys for storage system

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

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

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

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

This key *overrides* the default *"stored keys"* of the *optional 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 system"* and must be passed between braces. For example, if we execute at a second level:

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

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

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

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

6.1.1 Keys for `label` and `ref`

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

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

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

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

6.1.2   Keys for `wrap` and `display`

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

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

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

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

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

Displays the *argument* passed to the `\anskey`, the *body* for `anskey*` environment, the ⟨*label*⟩ for `\item*` and `\anspic*` at the place where it is executed. If the *optional argument* is present in `\item*` or `\anspic*` it will be shown using `wrap-opt` key.

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

Sets the *symbol* to be displayed in the left margin for `\anskey`, `anskey*`, `\item*` and `\anspic*` in the place where they are executed when using the key `show-ans`.

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

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

6.1.3   Keys for debug and checking

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

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

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

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

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

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

6.2   The command `\anskey`

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

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

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

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

6.2.1   Keys for `\anskey`

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

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

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

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

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

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

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

item-sym* = {⟨*symbol*⟩} *default: $\star$*

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

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

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

Example

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

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

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

6.3 The environment anskey*

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

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

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

The `anskey*` environment is implemented using the `scontents` package, for the correct operation `\begin{anskey*}` and `\end{anskey*}` must be in different lines, all ⟨*keys*⟩ must be passed separated by commas and "without separation" of the start of the environment. Comments "%" or "any character" after `\begin{anskey*}` or `[`⟨*key = val*⟩`]` on the same line are NOT supported, the package `scontents` will return an "error" message if this happens. In a similar way comments "%" or "any character" after `\end{anskey*}` on the same line the package `scontents` will return a "warning" message.

6.3.1 Keys for anskey*

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

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

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

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

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

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

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

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

Example

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

```
    \item Text containing our instructions or questions.
      \begin{enumext}
        \item Question.
          \begin{anskey*}
            ⟨second answer⟩
          \end{anskey*}
      \end{enumext}
    \item Text containing our instructions or questions.
      \begin{anskey*}
        ⟨third answer⟩
      \end{anskey*}
    \item Text containing our instructions or questions.
      \begin{anskey*}
        ⟨fourth answer⟩
      \end{anskey*}
\end{enumext}
```

⋆ 5. Text containing our instructions or questions.

[5] First answer with verbatim

6. Text containing our instructions or questions.

   (a) Question.

    [6] second answer

7. Text containing our instructions or questions.

[7] third answer

8. Text containing our instructions or questions.

[8] fourth answer

6.4 The environments keyans and keyans*

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

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

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

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

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

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

6.4.1 The \item* in keyans and keyans*

\item* `\item*`

        `\item*[⟨content⟩]`

The \item* and \item*[⟨content⟩] command *"store"* the current ⟨label⟩ set by label key next to the *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.

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

Example

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

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

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

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

Some text

6.5 The environment keyanspic

keyanspic `\begin{keyanspic}*[⟨n° upper, n° lower⟩]\anspic{⟨drawing⟩}\anspic*[⟨content⟩]{⟨drawing or tabular⟩}`

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

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

When the keyanspic environment is used *without arguments* the ⟨labels⟩ are centered *below* the drawings or tabular in a *single line* layout. The *starred argument* '*' places ⟨labels⟩ centered *above* the drawings or tabular.

The *optional argument* determines the number drawings or tabular placed at *upper and lower* in the environment. If the *optional argument* or the ⟨n° lower⟩ is omitted the drawings or tabular will be put on a *single line*. The vertical separation between *"upper"* and *"lower"* part is controlled by the values set by parsep key passed to keyans environment.

6.5.1 The command \anspic

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

The \anspic command take three arguments, the *starred argument* '*' store the current ⟨label⟩ next to the *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,nosep]
  \item Question with images.
    \begin{keyanspic}[3,2]
      \anspic{\includegraphics[scale=0.15]{example-image-a}}
      \anspic{\includegraphics[scale=0.15]{example-image-b}}
      \anspic{\includegraphics[scale=0.15]{example-image-a}}
      \anspic{\includegraphics[scale=0.15]{example-image-a}}
      \anspic*[note]{\includegraphics[scale=0.15]{example-image-a}}
    \end{keyanspic}
\end{enumext}
```

1. Question with images.
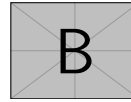
A) B) C)

D) * E)[note]

6.6 Printing stored content

6.6.1 The command \getkeyans

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

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

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

6.6.2 The command \foreachkeyans

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

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

Options for command

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

step = {⟨*integer*⟩}   default: 1
Sets the increment (⟨*step*⟩) applied to the value set by key start for each element stored in *prop list* {⟨*store name*⟩}. The value must be a ⟨*positive integer*⟩.

start = {⟨*integer*⟩}   default: 1
Sets the ⟨*position*⟩ of the *prop list* {⟨*store name*⟩} from which execution will start. The value must be a ⟨*positive integer*⟩.

stop = {⟨*integer*⟩}   default: 0
Sets the ⟨*position*⟩ of the *prop list* {⟨*store name*⟩} from which execution it will finish executing. The value must be a ⟨*positive integer*⟩.

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

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

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

6.6.3 The command \printkeyans

\printkeyans \printkeyans{⟨*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 environment by default or the enumext* environment if the *starred argument* '*' is used.

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

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

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

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

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

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

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

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

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

Example

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

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

  \item Related to Linux

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

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

\printkeyans{sample}
```

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

[1] $3(x + y + z)$

2. True False

(a) LaTeX2e is cool?

[2] Very True!

3. Related to Linux

(a) You use linux?

[3] Yes

(b) Rate the following package and class

i. `xsim`

[4] very good

ii. `exsheets`

[5] obsolete

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

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

7 Full examples

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

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

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

Example 1

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

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

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

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

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

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

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

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

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

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

Example 2

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

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

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

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

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

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

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

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

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

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

---

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

Example 3

A *"simple multiple choice"* test 📄.

1. First type of questions
   - Ⓐ value
   - Ⓑ correct
   - Ⓒ value
   - Ⓓ value
2. Second type of questions
   - I. $2\alpha + 2\delta = 90°$
   - II. $\alpha = \delta$
   - III. $\angle EDF = 45°$
   - Ⓐ I only
   - Ⓑ II only
   - Ⓒ I and II only
   - Ⓓ I and III only
   - Ⓔ I, II, and III
3. Third type of questions
   - (1) $2\alpha + 2\delta = 90°$
   - (2) $\angle EDF = 45°$
   - Ⓐ value
   - Ⓑ value
   - Ⓒ value
   - Ⓓ value
   - Ⓔ value
4. Question with image and label below:

   Ⓐ

   Ⓑ

   Ⓒ

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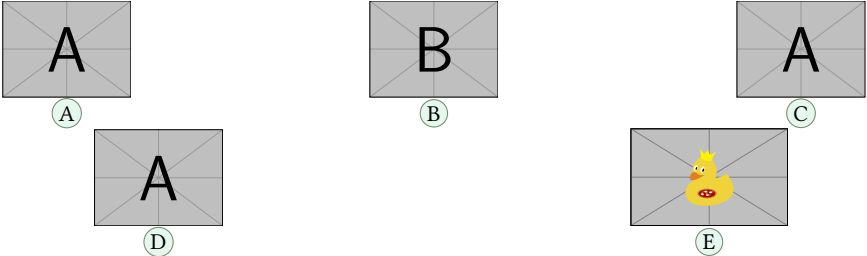
   Ⓔ
5. Question with image on left side:
   - Ⓐ value
   - Ⓑ value
   - Ⓒ value
   - Ⓓ correct
   - Ⓔ value
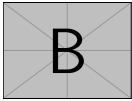
Test keys

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

Example 4

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

🦆 1. Factor $x^2 - 2x + 1$

🦆 2. Factor $3x + 3y + 3z$

The following questions need to be cuaqtified :)

🦆 3. True False
   - (a) $\alpha > \delta$
   - (b) LaTeX2e is cool?

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

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

1. $(x - 1)^2$
2. $3(x + y + z)$
3. (a) False
   (b) Very True!
4. (a) Yes
   (b) Yes, dnf
   (c) i. doesn't exist for now :(
       ii. very good
       iii. obsolete

Example 5

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

1

Which choice best describes what happens in the passage?

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

2

Which choice best describes what happens in the passage?

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

3

Which choice best describes what happens in the passage?

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

4

Which choice best describes what happens in the passage?

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

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

8 The way of non-enumerated lists

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

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

Fake `itemize` environment

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

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

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

Fake `description` environment

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

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

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

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

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

Description indented by label

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

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

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

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

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

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

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

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

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

Description with multi-line labels

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

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

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

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

Final notes

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

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

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

Why list environments?

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

Why not random questions and other utilities

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

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.

9 References

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

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

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

[4] González, Pablo. "scontents - Stores LaTeX contents in memory or files". Available from CTAN, https://www.ctan.org/pkg/scontents, 2022.

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

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

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

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

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

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

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

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

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

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

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

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

10   Change history

v1.0 2024-09-29   – First public release.

11 Index of Documentation

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

12   Implementation

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

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

12.1   General conventions

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

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

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

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

12.2   Initial set up

Start the DocStrip guards.

```
1  ⟨∗package⟩
```

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

```
2  ⟨@@=enumext⟩
```

12.3   Declaration of the package

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

```
3  \NeedsTeXFormat{LaTeX2e}[2024-06-01]
```

Now declare the enumext package.

```
4  \ProvidesExplPackage
5    {enumext}
6    {2024-09-29}
7    {1.0}
8    {Enumerate exercise sheets}
```

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

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

12.4   Definition of variables

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

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

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

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

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

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

Internal variables used by functions `\__enumext_is_not_nested:`, `\__enumext_is_on_first_level:` and `\__enumext_keyans_name_and_start:` (§12.5.1).

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

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

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

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

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

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

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

`\c__enumext_counter_style_tl`
`\l__enumext_ref_key_arg_tl`
`\l__enumext_ref_the_count_tl`
`\l__enumext_the_counter_X_tl`
`\l__enumext_renew_the_count_X_tl`

Internal variables used by ref key (§12.13).

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

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

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

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

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

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

`\l__enumext_current_widest_dim`
`\g__enumext_counter_styles_tl`
`\g__enumext_widest_label_tl`
`\l__enumext_label_width_by_box`

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

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

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

\l__enumext_leftmargin_tmp_X_bool
\l__enumext_leftmargin_tmp_X_dim
\l__enumext_leftmargin_X_dim
\l__enumext_itemindent_X_dim

The boolean variable \l__enumext_leftmargin_tmp_X_bool and the dimensional variable \l__enumext_-leftmargin_tmp_X_dim are used by the list-indent key (§12.17). The variables \l__enumext_-leftmargin_X_dim and \l__enumext_itemindent_X_dim are used and set by the function __enumext_-calc_hspace:NNNNNNNNNNNN (§12.37.1).

```
71 \cs_set_protected:Npn \__enumext_tmp:n #1
72   {
73     \bool_new:c { l__enumext_leftmargin_tmp_#1_bool }
74     \dim_new:c  { l__enumext_leftmargin_tmp_#1_dim  }
75     \dim_new:c  { l__enumext_leftmargin_#1_dim       }
76     \dim_new:c  { l__enumext_itemindent_#1_dim       }
77   }
78 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }
```

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

\l__enumext_multicols_above_X_skip
\l__enumext_multicols_below_X_skip
\g__enumext_multicols_right_X_skip
\l__enumext_align_label_pos_X_str

Internal variables used by columns key (§12.21) and align key (§12.12).

```
79 \cs_set_protected:Npn \__enumext_tmp:n #1
80   {
81     \skip_new:c  { l__enumext_multicols_above_#1_skip }
82     \skip_new:c  { l__enumext_multicols_below_#1_skip }
83     \skip_new:c  { g__enumext_multicols_right_#1_skip }
84     \str_new:c   { l__enumext_align_label_pos_#1_str  }
85   }
86 \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 (§12.22.4) and the keys mini-right, mini-right*, mini-env and mini-sep (§12.20, §12.22).

```
87 \int_new:N  \g__enumext_minipage_stat_int
88 \skip_new:N \l__enumext_minipage_temp_skip
89 \skip_new:N \l__enumext_minipage_left_skip
90 \skip_new:N \l__enumext_minipage_right_skip
91 \skip_new:N \l__enumext_minipage_after_skip
92 \skip_new:N \g__enumext_minipage_right_skip
93 \skip_new:N \g__enumext_minipage_after_skip
94 \cs_set_protected:Npn \__enumext_tmp:n #1
95   {
96     \dim_new:c  { l__enumext_minipage_left_#1_dim    }
97     \bool_new:c { l__enumext_minipage_active_#1_bool }
98   }
99 \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }
```

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

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

The bool vars \l__enumext_wrap_label_X_bool and \l__enumext_wrap_label_opt_X_bool are used by wrap-label and wrap-label* keys (§12.12), the integer \l__enumext_start_X_int are used by the start and start* keys (§12.14), the token list \l__enumext_fake_item_indent_X_tl is used by itemindent key (§12.17.1), the variables \l__enumext_label_fill_left_X_tl and \l__enumext_-label_fill_left_X_tl are used by the align key (§12.12). The boolean vars \l__enumext_vspace_-a_star_X_bool, \l__enumext_vspace_b_star_X_bool are used by above, above*, below and below* keys (§12.19).

```
100 \cs_set_protected:Npn \__enumext_tmp:n #1
101   {
102     \bool_new:c { l__enumext_wrap_label_#1_bool      }
103     \bool_new:c { l__enumext_wrap_label_opt_#1_bool }
104     \int_new:c  { l__enumext_start_#1_int            }
105     \tl_new:c   { l__enumext_fake_item_indent_#1_tl }
106     \tl_new:c   { l__enumext_label_fill_left_#1_tl  }
107     \tl_new:c   { l__enumext_label_fill_right_#1_tl }
108     \bool_new:c { l__enumext_vspace_a_star_#1_bool  }
109     \bool_new:c { l__enumext_vspace_b_star_#1_bool  }
110   }
111 \clist_map_inline:nn { i, ii, iii, iv, v, vii, viii } { \__enumext_tmp:n {#1} }
```

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

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

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

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

The variables `\l__enumext_store_current_label_tl` and `\l__enumext_store_current_opt_arg_-tl` save the *current label* and *optional argument* of `\item*` (§12.36) and `\anspic*` (§12.41.2) 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.

```
112 \bool_new:N \l__enumext_store_active_bool
113 \tl_new:N   \l__enumext_store_name_tl
114 \tl_new:N   \g__enumext_store_name_tl
115 \tl_new:N   \l__enumext_store_anskey_arg_tl
116 \tl_new:N   \l__enumext_store_anskey_env_tl
117 \tl_new:N   \l__enumext_store_anskey_opt_tl
118 \tl_new:N   \l__enumext_store_current_label_tl
119 \tl_new:N   \l__enumext_store_current_opt_arg_tl
120 \tl_new:N   \l__enumext_store_current_label_tmp_tl
```

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

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

```
121 \tl_new:N  \l__enumext_setkey_tmpa_tl
122 \tl_new:N  \l__enumext_setkey_tmpb_tl
123 \int_new:N \l__enumext_setkey_tmpa_int
124 \seq_new:N \l__enumext_setkey_tmpa_seq
125 \seq_new:N \l__enumext_setkey_tmpb_seq
```

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

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

```
126 \tl_new:N  \l__enumext_meta_path_tl
127 \seq_new:N \l__enumext_foreach_print_seq
128 \tl_new:N  \l__enumext_foreach_name_prop_tl
129 \tl_new:N  \g__enumext_foreach_default_keys_tl
```

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

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

```
130 \tl_new:N  \l__enumext_print_keyans_starred_tl
131 \str_new:N \l__enumext_mark_position_str
132 \tl_new:N  \g__enumext_item_symbol_aux_tl
133 \cs_set_protected:Npn \__enumext_tmp:n #1
134   {
135     \tl_new:c   { l__enumext_print_keyans_#1_tl         }
136     \tl_new:c   { l__enumext_store_save_key_#1_tl       }
137     \bool_new:c { l__enumext_store_save_key_#1_bool     }
138     \bool_new:c { l__enumext_store_upper_level_#1_bool }
139   }
140 \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {#1} }
```

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

Internal variables used by keyanspic environment and `\anspic` command (§12.41.1).

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

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

Internal variables used by *"internal check answer"* mechanism (§12.25.3) used by the check-ans and no-store keys and check for starred commands \item* in keyans and keyans* environments and \anspic* in keyanspic environment.

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

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

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

```
160 \bool_new:N \l__enumext_hyperref_bool
161 \bool_new:N \l__enumext_footnotes_key_bool
```

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

Internal variables used by save-ref key (§12.26). The variables \l__enumext_label_copy_X_tl correspond to temporary copies of the ⟨*labels*⟩ defined by level on which operations will be performed.

The variables \l__enumext_newlabel_arg_one_tl and \l__enumext_newlabel_arg_two_tl will be used to form the arguments passed to the function __enumext_newlabel:nn (§12.8) and the variable \l__enumext_write_aux_file_tl will be in charge of executing the writing code in the .aux file.

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

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

Internal variables used for redefinition of \footnote (§12.42.4).

```
170 \int_new:N \g__enumext_footnote_int
171 \seq_new:N \g__enumext_footnote_arg_seq
172 \seq_new:N \g__enumext_footnote_int_seq
```

(*End of definition for* \g__enumext_footnote_int, \g__enumext_footnote_arg_seq, *and* \g__enumext_footnote_int_seq.)

Internal variables used by enumext* and keyans* environments.

```
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`.)

12.5 Some utility functions

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

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

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

(*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.

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

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

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

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

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

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

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

(*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.

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

(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.

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

(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.

```
237  \cs_new_protected:Npn \__enumext_unskip_unkern:
238    {
239      \int_case:nnT { \lastnodetype }
240        {
241          { 11 }
242            {
243              % \typeout{SKIP} \typeout{\the\lastskip}
244              \unskip
245            }
246          { 12 }
247            {
248              % \typeout{KERN} \typeout{\the\lastkern}
249              \unkern
250            }
251        }
252    }
```

(End of definition for __enumext_unskip_unkern:.)

12.5.1 Utilities for environments and levels

__enumext_is_not_nested:
__enumext_is_on_first_level:

The function __enumext_is_not_nested: set the variables \g__enumext_standar_bool and \g__-enumext_starred_bool to *"true"* only if the environments enumext and enumext* are nested in each other and save the environment name in \l__enumext_envir_name_tl.

```
253  \cs_new_protected:Nn \__enumext_is_not_nested:
254    {
255      \str_case:en { \@currenvir }
256        {
257          {enumext}
258            {
259              \tl_set:Nn \l__enumext_envir_name_tl { enumext }
260              \bool_lazy_and:nnT
261                { \bool_not_p:n { \g__enumext_standar_bool } }
262                { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
263                {
264                  \bool_gset_true:N \g__enumext_standar_bool
265                }
266            }
267          {enumext*}
268            {
269              \tl_set:Nn \l__enumext_envir_name_tl { enumext* }
270              \bool_lazy_and:nnT
271                { \bool_not_p:n { \g__enumext_starred_bool } }
272                { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
273                {
274                  \bool_gset_true:N \g__enumext_starred_bool
275                }
276            }
277        }
278    }
```

The function __enumext_is_on_first_level: will set the variables \l__enumext_standar_first_-bool (§12.25.1), \l__enumext_starred_first_bool (§12.25.1) and \l__enumext_anskey_env_bool (§12.30) to *"true"* only if the environment is not nested and we are in the *"first level"* of it . We will also save the *start line number* of each environment in the variable \g__enumext_start_line_tl and the *name* of each environment in the variable \g__enumext_envir_name_tl to use in messages related to the check-ans key and .log file.

```
279  \cs_new_protected:Nn \__enumext_is_on_first_level:
280    {
281      \bool_lazy_all:nT
282        {
283          { \bool_if_p:N \g__enumext_standar_bool }
284          { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
285          { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
286        }
287        {
288          \bool_set_true:N \l__enumext_standar_first_bool
289          \bool_set_true:N \l__enumext_anskey_env_bool
290          \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
291          \tl_gset:Ne \g__enumext_start_line_tl
292            {
293              on ~ line ~ \exp_not:V \inputlineno
294            }
295        }
296      \bool_lazy_all:nT
297        {
298          { \bool_if_p:N \g__enumext_starred_bool }
299          { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
300          { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
301        }
302        {
303          \bool_set_true:N \l__enumext_starred_first_bool
304          \bool_set_true:N \l__enumext_anskey_env_bool
305          \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
306          \tl_gset:Ne \g__enumext_start_line_tl
307            {
308              on ~ line ~ \exp_not:V \inputlineno
309            }
310        }
311    }
```

(*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.

```
312  \cs_new_protected:Nn \__enumext_keyans_name_and_start:
313    {
314      \str_case:en { \@currenvir }
315        {
316          {keyans}
317            {
318              \tl_set:Nn \l__enumext_envir_name_tl { keyans }
319              \tl_set:Ne \l__enumext_check_start_line_env_tl
320                {
321                  in ~ 'keyans' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
322                }
323            }
324          {keyans*}
325            {
326              \tl_set:Nn \l__enumext_envir_name_tl { keyans* }
327              \tl_set:Ne \l__enumext_check_start_line_env_tl
328                {
329                  in ~ 'keyans*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
330                }
331            }
332          {keyanspic}
333            {
334              \tl_set:Nn \l__enumext_envir_name_tl { keyanspic }
335              \tl_set:Ne \l__enumext_check_start_line_env_tl
336                {
337                  in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
338                }
339            }
340        }
341    }
```

(*End of definition for* __enumext_keyans_name_and_start:.)

12.5.2 Utilities for log and terminal

__enumext_reset_global_vars:
__enumext_reset_global_int:
__enumext_reset_global_bool:
__enumext_reset_global_tl:

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

```
342 \cs_new_protected:Nn \__enumext_reset_global_vars:
343   {
344     \__enumext_reset_global_int:
345     \__enumext_reset_global_bool:
346     \__enumext_reset_global_tl:
347   }
348 \cs_new_protected:Nn \__enumext_reset_global_int:
349   {
350     \int_gzero:N \g__enumext_item_number_int
351     \int_gzero:N \g__enumext_item_anskey_int
352     \int_gzero:N \g__enumext_item_answer_diff_int
353   }
354 \cs_new_protected:Nn \__enumext_reset_global_bool:
355   {
356     \bool_gset_false:N \g__enumext_check_ans_key_bool
357     \bool_gset_false:N \g__enumext_standar_bool
358     \bool_gset_false:N \g__enumext_starred_bool
359   }
360 \cs_new_protected:Nn \__enumext_reset_global_tl:
361   {
362     \tl_gclear:N \g__enumext_store_name_tl
363     \tl_gclear:N \g__enumext_start_line_tl
364     \tl_gclear:N \g__enumext_envir_name_tl
365   }
```

(*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.

```
366 \cs_new_protected:Nn \__enumext_log_global_vars:
367   {
368     \msg_log:nneeee { enumext } { prop-seq-int-hook }
369       { \g__enumext_store_name_tl }
370       { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
371       { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
372       { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
373   }
```

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.

```
374 \cs_new_protected:Nn \__enumext_log_answer_vars:
375   {
376     \msg_log:nneee { enumext } { item-answer-hook }
377       { \int_use:N \g__enumext_item_number_int }
378       { \int_use:N \g__enumext_item_anskey_int }
379       { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int} }
380   }
```

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

12.6 Copying list and minipage environments

The list environment provided by LaTeX has the following plain form:

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

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

🌐 For compatibility with LaTeX *tagged* PDF we should use \NewCommandCopy and not \cs_new_eq:NN. When *tagged* PDF is active \item is redefined using ltcmd (see latex-lab-block).

__enumext_start_list:nn
__enumext_stop_list:
__enumext_item_std:w

The functions __enumext_start_list:nn, __enumext_stop_list: and __enumext_item_std:w correspond to copies of \list, \endlist and \item from plain definition of list environment.

```
381 \__enumext_at_begin_document:n
```

```
382   {
383     \cs_new_eq:NN   \__enumext_start_list:nn \list
384     \cs_new_eq:NN   \__enumext_stop_list: \endlist
385     \NewCommandCopy \__enumext_item_std:w \item
386   }
```

(*End of definition for __enumext_start_list:nn, __enumext_stop_list:, and __enumext_item_std:w.*)

The `minipage` environment provided by LaTeX has the following (simplified) plain form:

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

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

__enumext_minipage:w
__enumext_endminipage:

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

```
387 \__enumext_at_begin_document:n
388   {
389     \cs_new_eq:NN \__enumext_minipage:w \minipage
390     \cs_new_eq:NN \__enumext_endminipage: \endminipage
391   }
```

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

12.7 The internal `minipage` environment

__enumext_internal_mini_page:
__enumext_mini_env*

The function `\__enumext_internal_mini_page:` creates a internal `__enumext_mini_page` environment (*custom version* of `minipage`) setting the `\if@minipage` switch to *"false"* to allow spaces at the *"above"* of the environment, plus we will add `\skip_vertical:N \c_zero_skip` to maintain alignment on *"top"* in the first part and `\skip_vertical:N \c_zero_skip` in the second part to allow spaces *"below"*. This environment will be used internally by the `mini-env` key, it is not documented in the user interface and is for internal use only. This function is passed to the function `\__enumext_safe_exec:` in the `enumext` environment definition (§12.38) and `\__enumext_safe_exec_vii:` in the `enumext*` environment definition (§12.43)

```
392 \cs_new_protected:Nn \__enumext_internal_mini_page:
393   {
394     \int_compare:nNnT { \l__enumext_level_int } = { 0 }
395       {
396         \DeclareDocumentEnvironment{__enumext_mini_page}{ m }
397           {
398             \__enumext_minipage:w [ t ] { ##1 }
399               \legacy_if_gset_false:n { @minipage }
400               \skip_vertical:N \c_zero_skip
401           }
402           {
403               \skip_vertical:N \c_zero_skip
404             \__enumext_endminipage:
405           }
406       }
407   }
```

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

12.8 Compatibility with hyperref and footnotehyper

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

```
408 \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
409 \hook_gset_rule:nnnn { begindocument } { enumext } { after } { hyperref }
```

__enumext_after_hyperref:
__enumext_hypertarget:nn
__enumext_phantomsection:

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

```
410 \cs_new_protected:Nn \__enumext_after_hyperref:
411   {
412     \IfPackageLoadedTF { hyperref }
413       {
414         \msg_info:nnn { enumext } { package-load } { hyperref }
415         \bool_set_true:N \l__enumext_hyperref_bool
```

```
416          \IfHyperBoolean{hyperfootnotes}
417            {
418              % \typeout{hyperfootnotes=true}
419              \bool_set_true:N \l__enumext_footnotes_key_bool
420            }
421            {
422              % \typeout{hyperfootnotes=false}
423            }
424          }
425        {  }
```

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.

```
426      \bool_if:NT \l__enumext_footnotes_key_bool
427        {
428          \IfPackageLoadedTF { footnotehyper }
429            {
430              \msg_info:nnn { enumext } { package-load } { footnotehyper }
431            }
432            {
433              % \typeout{No ~ footnotehyper ~ load}
434              % \typeout{Load ~ and  ~ use  ~ \string\makesavenoteenv{enumext*}}
435              \bool_set_false:N \l__enumext_footnotes_key_bool
436            }
437        }
```

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.

```
438      \bool_if:NTF \l__enumext_hyperref_bool
439        {
440          \cs_new_eq:NN \__enumext_hypertarget:nn \hypertarget
441          \cs_new_eq:NN \__enumext_phantomsection: \phantomsection
442        }
443        {
444          \cs_new_eq:NN \__enumext_hypertarget:nn \use_none:nn
445          \cs_new_eq:NN \__enumext_phantomsection: \prg_do_nothing:
446        }
447      }
```

(*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.

```
448  \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
449    {
450      \protected@write \@auxout { }
451        {
452          \token_to_str:N \newlabel {#1}
453            {
454              {#2}
455              \bool_if:NT \l__enumext_hyperref_bool
456                { { \thepage } {#2} {#1} }
457                { }
458            }
459        }
460      \__enumext_hypertarget:nn {#1} { }
461      \__enumext_phantomsection:
462    }
```

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

12.9 Definition of public dimension

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

```
463 \dim_zero_new:N \itemwidth
```

12.10 Definition of counters

__enumext_define_counters:Nn
__enumext_define_counters:cn

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

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

#2 : The counter's name.

```
464 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
465   {
466     \cs_if_exist:cTF { c@ #2 }
467       { \msg_fatal:nnn { enumext } { counters }{ #2 } }
468       {
469         \tl_set:Nn #1 { #2 }
470         \newcounter { #2 }
471       }
472   }
```

(End of definition for __enumext_define_counters:Nn.)

enumXi
enumXii
enumXiii
enumXiv
enumXv
enumXvi
enumXvii
enumXviii

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

```
473 \__enumext_define_counters:Nn \l__enumext_counter_i_tl    { enumXi     }
474 \__enumext_define_counters:Nn \l__enumext_counter_ii_tl   { enumXii    }
475 \__enumext_define_counters:Nn \l__enumext_counter_iii_tl  { enumXiii   }
476 \__enumext_define_counters:Nn \l__enumext_counter_iv_tl   { enumXiv    }
477 \__enumext_define_counters:Nn \l__enumext_counter_v_tl    { enumXv     }
478 \__enumext_define_counters:Nn \l__enumext_counter_vi_tl   { enumXvi    }
479 \__enumext_define_counters:Nn \l__enumext_counter_vii_tl  { enumXvii   }
480 \__enumext_define_counters:Nn \l__enumext_counter_viii_tl { enumXviii  }
```

(End of definition for enumXi and others.)

12.11 Definition of labels

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

__enumext_register_counter_style:Nn

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

```
481 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
482   {
483     \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
484     \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
485   }
486 \__enumext_register_counter_style:Nn \arabic { 0 }
487 \__enumext_register_counter_style:Nn \Alph   { M }
488 \__enumext_register_counter_style:Nn \alph   { m }
489 \__enumext_register_counter_style:Nn \Roman  { VIII }
490 \__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.

```
491 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
492   {
493     \hbox_set:Nn \l__enumext_label_width_by_box {#2}
494     \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
495   }
496 \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.

```
497 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
498   {
499     \tl_clear_new:N #1
500     \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
501     \tl_gset_eq:NN \g__enumext_widest_label_tl #1
502     \tl_map_inline:Nn \g__enumext_counter_styles_tl
503       {
504         \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
505         \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
506           { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
507       }
508     \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
509       { \tl_use:N \g__enumext_widest_label_tl }
510     \tl_set_eq:cN { the #2 } #1
511   }
512 \cs_generate_variant:Nn \__enumext_label_style:Nnn { cvn }
```

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

12.12 Setting keys associated with label

font
labelsep
labelwidth
wrap-label
wrap-label*

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

```
513 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
514   {
515     \keys_define:nn { enumext / #1 }
516       {
517         font        .tl_set:c   = { l__enumext_label_font_style_#2_tl },
518         font        .value_required:n = true,
519         labelsep    .dim_set:c  = { l__enumext_labelsep_#2_dim },
520         labelsep    .initial:n  = {0.3333em},
521         labelsep    .value_required:n = true,
522         labelwidth  .dim_set:c  = { l__enumext_labelwidth_#2_dim },
523         labelwidth  .value_required:n = true,
524         wrap-label  .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
525         wrap-label  .initial:n  = {##1},
526         wrap-label  .value_required:n = true,
527         wrap-label* .code:n = {
528                               \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
529                               \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
530                              },
531         wrap-label* .value_required:n = true,
532       }
533   }
534 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

🧨 In this point, the following are set __enumext_wrapper_label_X:n which will be used by __enumext_make_label: for the different levels of the enumext environment and is set to __enumext_wrapper_label_v:n which will be used by __enumext_keyans_make_label: for keyans and keyanspic environments.

align

The align key is implemented differently for *"starred"* and *"non starred"* environments.

```
535 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
536   {
537     \keys_define:nn { enumext / #1 }
538       {
539         align .choice:,
540         align / left    .code:n =
541                         {
542                           \tl_clear:c { l__enumext_label_fill_left_#2_tl  }
543                           \tl_set:cn  { l__enumext_label_fill_right_#2_tl } { \hfill }
544                           \str_set:cn { l__enumext_align_label_pos_#2_str } { l }
545                         },
546         align / right   .code:n =
547                         {
548                           \tl_set:cn  { l__enumext_label_fill_left_#2_tl  } { \hfill }
```

```
549                              \tl_clear:c { l__enumext_label_fill_right_#2_tl }
550                              \str_set:cn { l__enumext_align_label_pos_#2_str } { r }
551                            },
552          align / center  .code:n =
553                            {
554                              \tl_set:cn { l__enumext_label_fill_left_#2_tl  } { \hfill }
555                              \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
556                              \str_set:cn { l__enumext_align_label_pos_#2_str } { c }
557                            },
558          align / unknown .code:n =
559                            \msg_error:nneee { enumext } { unknown-choice }
560                              { align } { left, ~ right, ~  center } { \exp_not:n {##1} },
561          align .initial:n  = left,
562          align .value_required:n  = true,
563        }
564    }
565  \clist_map_inline:nn
566    {
567      {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
568    }
569    { \__enumext_tmp:nn #1 }
```

🔹 For compatibility with LaTeX *tagged* PDF we must set `\l__enumext_align_label_pos_X_str`. When *tagged* PDF is active `\makelabel` is redefined and the only way to get the `align` key to work correctly is by using `\makebox`.

```
570  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
571    {
572      \keys_define:nn { enumext / #1 }
573        {
574          align .choice:,
575          align / left    .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
576          align / right   .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
577          align / center  .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
578          align / unknown .code:n =
579                            \msg_error:nneee { enumext } { unknown-choice }
580                              { align } { left, ~ right, ~  center } { \exp_not:n {##1} },
581          align .initial:n = left,
582          align .value_required:n = true,
583        }
584    }
585  \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }
```

(*End of definition for* `align`.)

12.13   Setting `label` and `ref` keys

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

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

label
ref
\l__enumext_label_i_tl
\l__enumext_label_ii_tl
\l__enumext_label_iii_tl
\l__enumext_label_iv_tl

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

```
586  \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
587    {
588      \keys_define:nn { enumext / #1 }
589        {
590          label .code:n    = {
591                              \__enumext_label_style:cvn { l__enumext_label_#2_tl }
592                                { l__enumext_counter_#2_tl } {##1}
593                              \dim_set_eq:cN  { l__enumext_labelwidth_#2_dim }
594                                \l__enumext_current_widest_dim
595                            },
596          label .initial:n = #3,
597          label .value_required:n = true,
598          ref   .code:n    = \__enumext_standar_ref:n {##1},
599          ref   .value_required:n = true,
600        }
601    }
602  \__enumext_tmp:nnn { level-1 } {   i } { \arabic*.}
603  \__enumext_tmp:nnn { level-2 } {  ii } { (\alph*) }
604  \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
605  \__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.

```
606 \cs_new_protected:Npn \__enumext_standar_ref:n #1
607   {
608     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
609     \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
610       {
611         \msg_error:nnn { enumext } { key-ref-empty } { enumext }
612       }
613       {
614         \tl_set_eq:Nc
615           \l__enumext_ref_the_count_tl { l__enumext_counter_ \__enumext_level: _tl }
616         \__enumext_regex_counter_style:
617         \tl_set_eq:Nc
618           \l__enumext_ref_the_count_tl { l__enumext_the_counter_ \__enumext_level: _tl }
619         \tl_put_right:ce { l__enumext_renew_the_count_ \__enumext_level: _tl }
620           {
621             \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
622               { \exp_not:V \l__enumext_ref_key_arg_tl }
623           }
624       }
625   }
```

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

```
626 \cs_new_protected:Nn \__enumext_standar_ref:
627   {
628     \tl_if_empty:cF { l__enumext_renew_the_count_ \__enumext_level: _tl }
629       {
630         \tl_use:c { l__enumext_renew_the_count_ \__enumext_level: _tl }
631       }
632   }
```

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

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

label
ref
\l__enumext_label_vii_tl
\l__enumext_label_viii_tl

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

```
633 \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
634   {
635     \keys_define:nn { enumext / #1 }
636       {
637         label .code:n    = {
638                             \__enumext_label_style:cvn { l__enumext_label_#2_tl }
639                               { l__enumext_counter_#2_tl } {##1}
640                             \dim_set_eq:cN  { l__enumext_labelwidth_#2_dim }
641                               \l__enumext_current_widest_dim
642                           },
643         label .initial:n = #3,
644         label .value_required:n = true,
645         ref   .code:n    = \__enumext_starred_ref:n {##1},
646         ref   .value_required:n = true,
647       }
648   }
649 \__enumext_tmp:nnn { enumext* } {  vii } { \arabic*.}
650 \__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.

```
651 \cs_new_protected:Npn \__enumext_starred_ref:n #1
652   {
653     \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
654     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
655       {
```

```
656            \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
657              {
658                \msg_error:nnn { enumext } { key-ref-empty } { enumext* }
659              }
660              {
661                \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_vii_tl
662                \__enumext_regex_counter_style:
663                \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_vii_tl
664                \tl_put_right:Ne \l__enumext_renew_the_count_vii_tl
665                  {
666                    \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
667                      { \exp_not:V \l__enumext_ref_key_arg_tl }
668                  }
669              }
670          }
671      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
672          {
673            \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
674              {
675                \msg_error:nnn { enumext } { key-ref-empty } { keyans* }
676              }
677              {
678                \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_viii_tl
679                \__enumext_regex_counter_style:
680                \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_viii_tl
681                \tl_put_right:Ne \l__enumext_renew_the_count_viii_tl
682                  {
683                    \exp_not:N \renewcommand  { \exp_not:V \l__enumext_ref_the_count_tl }
684                      { \exp_not:V \l__enumext_ref_key_arg_tl }
685                  }
686              }
687          }
688      }
```

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.

```
689  \cs_new_protected:Nn \__enumext_starred_ref:
690    {
691      \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
692        {
693          \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
694            {
695              \tl_use:N \l__enumext_renew_the_count_vii_tl
696            }
697        }
698      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
699        {
700          \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
701            {
702              \tl_use:N \l__enumext_renew_the_count_viii_tl
703            }
704        }
705    }
```

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

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

label
ref

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

`\l__enumext_label_v_tl`
`\l__enumext_label_vi_tl`

```
706  \keys_define:nn { enumext / keyans }
707    {
708      label .code:n    = {
709                          \__enumext_label_style:cvn { l__enumext_label_v_tl }
710                            { l__enumext_counter_v_tl } {#1}
711                          \dim_set_eq:cN  { l__enumext_labelwidth_v_dim }
712                            \l__enumext_current_widest_dim
713                          \__enumext_label_style:cvn { l__enumext_label_vi_tl }
714                            { l__enumext_counter_vi_tl } {#1}
715                          \dim_set_eq:cN  { l__enumext_labelwidth_v_dim }
716                            \l__enumext_current_widest_dim
717                        },
```

```
718    label .initial:n = \Alph*),
719    label .value_required:n = true,
720    ref    .code:n    = \__enumext_keyans_ref:n {#1},
721    ref    .value_required:n = true,
722  }
```

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

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

```
723  \cs_new_protected:Npn \__enumext_keyans_ref:n #1
724    {
725      \tl_set:Nn \l__enumext_ref_key_arg_tl {#1}
726      \tl_if_empty:NTF \l__enumext_ref_key_arg_tl
727        {
728          \msg_error:nnn { enumext } { key-ref-empty } { keyans }
729        }
730        {
731          \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_counter_v_tl
732          \__enumext_regex_counter_style:
733          \tl_set_eq:NN \l__enumext_ref_the_count_tl \l__enumext_the_counter_v_tl
734          \tl_put_right:Ne \l__enumext_renew_the_count_v_tl
735            {
736              \exp_not:N \renewcommand { \exp_not:V \l__enumext_ref_the_count_tl }
737                { \exp_not:V \l__enumext_ref_key_arg_tl }
738            }
739        }
740    }
```

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

```
741  \cs_new_protected:Nn \__enumext_keyans_ref:
742    {
743      \tl_if_empty:NF \l__enumext_renew_the_count_v_tl
744        {
745          \tl_use:N \l__enumext_renew_the_count_v_tl
746        }
747    }
```

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

12.14 Setting start, start* and widest keys

__enumext_start_from:NNn   The function __enumext_start_from:NNn used by start and start* keys take three arguments:
__enumext_start_from:ccn   #1:   \l__enumext_label_X_tl
__enumext_start_from:cce   #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.

```
748  \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
749    {
750      \__enumext_if_is_int:nTF { #3 }
751        {
752          \int_set:Nn #2 {#3}
753        }
754        {
755          \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
756            { \int_set:Nn #2 { \int_from_alph:n {#3} } }
757          \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
758            { \int_set:Nn #2  { \int_from_roman:n {#3} } }
759        }
760    }
761  \cs_generate_variant:Nn \__enumext_start_from:NNn { ccn, cce }
```

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

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

```
762  \cs_new_protected:Npn \__enumext_widest_from:nNNn #1 #2 #3 #4
763    {
764      \__enumext_if_is_int:nTF {#4}
765        {
766          \setcounter{enumX#1} { #4 }
767        }
768        {
769          \regex_match:nVT { \c{Alph} | \c{alph} } {#2}
770            { \setcounter{enumX#1} { \int_from_alph:n {#4} } }
771          \regex_match:nVT { \c{Roman} | \c{roman} } {#2}
772            { \setcounter{enumX#1} { \int_from_roman:n {#4} } }
773        }
774      \__enumext_label_width_by_box:cv
775        { l__enumext_labelwidth_#1_dim } { l__enumext_label_#1_tl }
776    }
777  \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.

```
778  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
779    {
780      \keys_define:nn { enumext / #1 }
781        {
782          start* .code:n    = {
783                              \__enumext_start_from:ccn
784                                { l__enumext_label_#2_tl }
785                                { l__enumext_start_#2_int } {##1}
786                              },
787          start* .value_required:n = true,
788          start  .code:n    = {
789                              \__enumext_start_from:cce
790                                { l__enumext_label_#2_tl }
791                                { l__enumext_start_#2_int } { \int_eval:n {##1} }
792                              },
793          start  .initial:n = 1,
794          start  .value_required:n = true,
795          widest .code:n    = {
796                              \__enumext_widest_from:nccn {#2}
797                                { l__enumext_label_#2_tl }
798                                { l__enumext_labelwidth_#2_dim } {##1}
799                              },
800          widest .value_required:n = true,
801        }
802    }
803  \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

12.15   Setting keys for vertical spaces

topsep
partopsep
parsep
noitemsep
nosep

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

```
804  \cs_set_protected:Npn \__enumext_tmp:nnnnnn #1 #2 #3 #4 #5 #6
805    {
806      \keys_define:nn { enumext / #1 }
807        {
808          topsep     .skip_set:c = { l__enumext_topsep_#2_skip },
809          topsep     .initial:n  = {#3},
810          topsep     .value_required:n = true,
811          partopsep .skip_set:c = { l__enumext_partopsep_#2_skip },
812          partopsep .initial:n  = {#4},
813          partopsep .value_required:n = true,
814          parsep     .skip_set:c = { l__enumext_parsep_#2_skip },
815          parsep     .initial:n  = {#5},
816          parsep     .value_required:n = true,
```

```
817          itemsep   .skip_set:c = { l__enumext_itemsep_#2_skip },
818          itemsep   .initial:n  = {#6},
819          itemsep   .value_required:n = true,
820          noitemsep .meta:n     = { itemsep = 0pt, parsep = 0pt },
821          noitemsep .value_forbidden:n = true,
822          nosep     .meta:n     = {
823                                  itemsep = 0pt, parsep= 0pt,
824                                  topsep = 0pt, partopsep = 0pt,
825                                },
826          nosep     .value_forbidden:n = true,
827        }
828    }
```

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

```
829 \__enumext_tmp:nnnnnn { level-1 } { i } { 8.0pt plus 2.0pt minus 4.0pt }
830   { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
831   { 4.0pt plus 2.0pt minus 1.0pt }
832 \__enumext_tmp:nnnnnn { level-2 } { ii } { 4.0pt plus 2.0pt minus 1.0pt }
833   { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
834   { 2.0pt plus 1.0pt minus 1.0pt }
835 \__enumext_tmp:nnnnnn { level-3 } { iii } { 2.0pt plus 1.0pt minus 1.0pt }
836   { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
837 \__enumext_tmp:nnnnnn { level-4 } { iv } { 2.0pt plus 1.0pt minus 1.0pt }
838   { 1.0pt minus 1.0pt }{ 0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
839 \__enumext_tmp:nnnnnn { keyans  } { v }{ 4.0pt plus 2.0pt minus 1.0pt }
840   { 2.0pt plus 1.0pt minus 1.0pt }{ 2.0pt plus 1.0pt minus 1.0pt }
841   { 2.0pt plus 1.0pt minus 1.0pt }
842 \__enumext_tmp:nnnnnn { enumext* } { vii } { 8.0pt plus 2.0pt minus 4.0pt }
843   { 2.0pt plus 1.0pt minus 1.0pt } { 4.0pt plus 2.0pt minus 1.0pt }
844   { 4.0pt plus 2.0pt minus 1.0pt }
845 \__enumext_tmp:nnnnnn { keyans* } { viii } { 4.0pt plus 2.0pt minus 1.0pt }
846   { 2.0pt plus 1.0pt minus 1.0pt } { 2.0pt plus 1.0pt minus 1.0pt }
847   { 2.0pt plus 1.0pt minus 1.0pt }
```

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

12.16   Setting base-fix key

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

base-fix

__enumext_nested_base_line_fix:

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

```
848 \cs_set_protected:Npn \__enumext_tmp:n #1
849   {
850     \keys_define:nn { enumext / #1 }
851       {
852         base-fix .bool_set:N = \l__enumext_base_line_fix_bool,
853         base-fix .initial:n  = false,
854         base-fix .value_forbidden:n = true,
855       }
856   }
857 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

The function __enumext_nested_base_line_fix: will be in charge of applying the baseline correction and adjusting the ⟨*keys*⟩. This function is passed to the function __enumext_parse_keys:n in the enumext environment definition (§12.38) and to the function __enumext_parse_keys_vii:n in the enumext* environment definition (§12.43)

🎯 This key is enabled by default in the command \printkeyans (§12.46).

```
858 \cs_new_protected:Nn \__enumext_nested_base_line_fix:
859   {
860     \bool_lazy_and:nnT
861       { \bool_if_p:N \l__enumext_standar_first_bool }
862       { \bool_if_p:N \l__enumext_base_line_fix_bool }
863       {
864         \mode_leave_vertical:
865         \vspace { -\baselineskip }
866         \keys_set:nn { enumext / level-1 }
867           {
868             topsep = 0pt, above = 0pt, above* = 0pt,
869           }
```

```
870          }
871      \bool_lazy_and:nnT
872        { \bool_if_p:N \l__enumext_starred_first_bool }
873        { \bool_if_p:N \l__enumext_base_line_fix_bool }
874        {
875          \mode_leave_vertical:
876          \vspace { -\baselineskip }
877          \keys_set:nn { enumext / enumext* }
878            {
879              topsep = 0pt, above = 0pt, above* = 0pt,
880            }
881        }
882      \bool_set_false:N \l__enumext_base_line_fix_bool
883    }
```

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

12.17 Setting keys for horizontal spaces

itemindent
rightmargin
listparindent
list-offset
list-indent

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

```
884 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
885   {
886     \keys_define:nn { enumext / #1 }
887       {
888         itemindent    .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
889         itemindent    .value_required:n = true,
890         rightmargin   .dim_set:c = { l__enumext_rightmargin_#2_dim },
891         rightmargin   .value_required:n = true,
892         listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
893         listparindent .value_required:n = true,
894         list-offset   .dim_set:c = { l__enumext_listoffset_#2_dim },
895         list-offset   .value_required:n = true,
896         list-indent   .code:n     =
897                          \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
898                          \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
899         list-indent   .value_required:n = true,
900       }
901   }
902 \clist_map_inline:nn
903   {
904     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
905   }
906   { \__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.

```
907 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
908   {
909     \keys_define:nn { enumext / #1 }
910       {
911         itemindent    .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
912         itemindent    .value_required:n = true,
913         rightmargin   .dim_set:c = { l__enumext_rightmargin_#2_dim },
914         rightmargin   .value_required:n = true,
915         listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
916         listparindent .value_required:n = true,
917         list-offset   .dim_set:c = { l__enumext_listoffset_#2_dim },
918         list-offset   .value_required:n = true,
919         list-indent   .meta:n     = { list-offset = ##1 },
920         list-indent   .value_required:n = true,
921       }
922   }
923 \clist_map_inline:nn
924   {
925     {enumext*}{vii}, {keyans*}{viii}
926   }
927   { \__enumext_tmp:nn #1 }
```

12.17.1 Functions for setting the fake `itemindent`

`\__enumext_fake_item_indent:`
`\__enumext_keyans_fake_item_indent:`
`\__enumext_fake_item_vii:`
`\__enumext_fake_item_viii:`

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

```
928  \cs_set_protected:Nn \__enumext_fake_item_indent:
929    {
930      \dim_compare:nNnT
931        { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
932        >
933        { \c_zero_dim }
934        {
935          \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
936            {
937              \exp_not:N \mode_leave_vertical:
938              \exp_not:n { \skip_horizontal:n }
939                { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
940              \ignorespaces
941            }
942        }
943    }
944  \cs_set_protected:Nn \__enumext_keyans_fake_item_indent:
945    {
946      \dim_compare:nNnT
947        { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
948        {
949          \tl_set:Ne \l__enumext_fake_item_indent_v_tl
950            {
951              \exp_not:N \mode_leave_vertical:
952              \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
953              \ignorespaces
954            }
955        }
956    }
957  \cs_set_protected:Nn \__enumext_fake_item_vii:
958    {
959      \dim_compare:nNnT
960        { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
961        {
962          \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
963            {
964              \exp_not:N \mode_leave_vertical:
965              \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
966              \ignorespaces
967            }
968        }
969    }
970  \cs_set_protected:Nn \__enumext_fake_item_viii:
971    {
972      \dim_compare:nNnT
973        { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
974        {
975          \tl_set:Ne \l__enumext_fake_item_indent_viii_tl
976            {
977              \exp_not:N \mode_leave_vertical:
978              \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
979              \ignorespaces
980            }
981        }
982    }
```

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

12.18 Setting `show-length` key

show-length

Define and set `show-length` key for `enumext`, `enumext*`, `keyans` and `keyans*` environments. The function sets the boolean variable `\l__enumext_show_length_X_bool` used in the definition of all environments to "*true*" and calls the function `\__enumext_show_length:nnn` which prints all the values of the "*vertical*" and "*horizontal*" parameters calculated and used.

```
983  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
984    {
```

```
985        \keys_define:nn { enumext / #1 }
986          {
987            show-length .bool_set:c = { l__enumext_show_length_#2_bool },
988            show-length .initial:n  = false,
989          }
990      }
991    \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

12.19 Setting before, after and first keys

before
before*
after
first

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

```
992    \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
993      {
994        \keys_define:nn { enumext / #1 }
995          {
996            before  .tl_set:c   = { l__enumext_before_no_starred_key_#2_tl },
997            before  .value_required:n = true,
998            before* .tl_set:c   = { l__enumext_before_starred_key_#2_tl },
999            before* .value_required:n = true,
1000           after   .tl_set:c   = { l__enumext_after_stop_list_#2_tl },
1001           after   .value_required:n = true,
1002           first   .tl_set:c   = { l__enumext_after_list_args_#2_tl },
1003           first   .value_required:n = true,
1004         }
1005     }
1006   \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

12.19.1 Functions for before, after and first keys in enumext

__enumext_before_args_exec:
__enumext_before_keys_exec:
__enumext_after_stop_list:
__enumext_after_args_exec:

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

```
1007   \cs_new_protected:Nn \__enumext_before_args_exec:
1008     {
1009       \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
1010     }
```

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*⟩}}

```
1011   \cs_new_protected:Nn \__enumext_before_keys_exec:
1012     {
1013       \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
1014     }
```

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

```
1015   \cs_new_protected:Nn \__enumext_after_stop_list:
1016     {
1017       \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
1018     }
```

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.

```
1019   \cs_new_protected:Nn \__enumext_after_args_exec:
1020     {
1021       \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
1022     }
```

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

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

__enumext_before_args_exec_v:

__enumext_before_keys_exec_v:

__enumext_after_stop_list_v:

__enumext_after_args_exec_v:

Same implementation as the one used in the enumext environment.

```
1023 \cs_new_protected:Nn \__enumext_before_args_exec_v:
1024   {
1025     \tl_use:N \l__enumext_before_starred_key_v_tl
1026   }
1027 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
1028   {
1029     \tl_use:N \l__enumext_before_no_starred_key_v_tl
1030   }
1031 \cs_new_protected:Nn \__enumext_after_stop_list_v:
1032   {
1033     \tl_use:N \l__enumext_after_stop_list_v_tl
1034   }
1035 \cs_new_protected:Nn \__enumext_after_args_exec_v:
1036   {
1037     \tl_use:N \l__enumext_after_list_args_v_tl
1038   }
```

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

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

__enumext_before_args_exec_vii:

__enumext_before_keys_exec_vii

__enumext_after_stop_list_vii:

__enumext_after_args_exec_vii:

Same implementation as the one used in the enumext environment.

```
1039 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
1040   {
1041     \tl_use:N \l__enumext_before_starred_key_vii_tl
1042   }
1043 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
1044   {
1045     \tl_use:N \l__enumext_before_starred_key_viii_tl
1046   }
1047 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
1048   {
1049     \tl_use:N \l__enumext_before_no_starred_key_vii_tl
1050   }
1051 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
1052   {
1053     \tl_use:N \l__enumext_before_no_starred_key_viii_tl
1054   }
1055 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
1056   {
1057     \tl_use:N \l__enumext_after_stop_list_vii_tl
1058   }
1059 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
1060   {
1061     \tl_use:N \l__enumext_after_stop_list_viii_tl
1062   }
1063 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
1064   {
1065     \tl_use:N \l__enumext_after_list_args_vii_tl
1066   }
1067 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
1068   {
1069     \tl_use:N \l__enumext_after_list_args_viii_tl
1070   }
```

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

12.20 Setting keys for `multicols` and `minipage`

mini-env

mini-sep

columns-sep

columns

The default value of the columns-sep key is handled by the state of the boolean variable \l__enumext_-columns_sep_X_bool which is handled in the internal definition of the enumext and keyans environments. Define and set mini-env, mini-sep, columns-sep and columns keys for enumext, enumext*, keyans and keyans* environments.

```
1071 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1072   {
1073     \keys_define:nn { enumext / #1 }
1074       {
1075         mini-env    .dim_set:c  = { l__enumext_minipage_right_#2_dim },
1076         mini-env    .value_required:n = true,
1077         mini-sep    .dim_set:c  = { l__enumext_minipage_hsep_#2_dim },
```

```
1078        mini-sep    .initial:n  = 0.3333em,
1079        mini-sep    .value_required:n = true,
1080        columns-sep .dim_set:c  = { l__enumext_columns_sep_#2_dim },
1081        columns-sep .value_required:n = true,
1082        columns     .int_set:c  = { l__enumext_columns_#2_int },
1083        columns     .initial:n  = 1,
1084        columns     .value_required:n = true,
1085      }
1086   }
1087 \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.

```
1088 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1089   {
1090     \keys_define:nn { enumext / #1 }
1091       {
1092         mini-right  .tl_gset:c = { g__enumext_miniright_code_#2_tl },
1093         mini-right  .value_required:n = true,
1094         mini-right* .code:n    = {
1095                                 \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
1096                                 \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1097                                 },
1098         mini-right* .value_required:n = true,
1099       }
1100   }
1101 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }
```

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

12.21   Adjustment of vertical spaces for `multicols`

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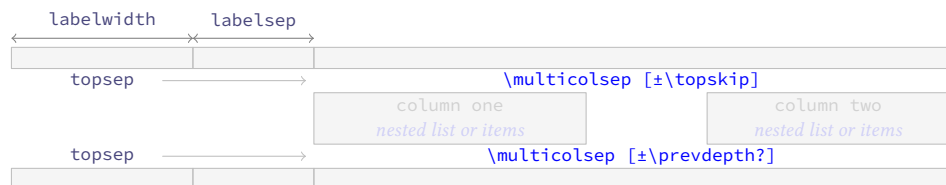
Figure 7: Representation of the vertical space in `multicols` for a nested level.

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

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

12.21.1   Adjustment of vertical spaces for `multicols` in enumext

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

We will set the default values taking into account that TeX is in ⟨*horizontal mode*⟩, then we will make the settings for the ⟨*vertical mode*⟩ in which `\partopsep` comes into play.

Set the values of `\l__enumext_multicols_above_X_skip` and `\l__enumext_multicols_below_X_-skip` equal to the value of `\topsep` in the *current level*.

```
1102 \cs_new_protected:Nn \__enumext_multi_set_vskip:
1103   {
1104     \skip_set:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1105       {
1106         \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1107       }
1108     \skip_set:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1109       {
1110         \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1111       }
```

```
1112    \__enumext_add_pre_parsep:
1113  }
```

(*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.*

```
1114  \cs_new_protected:Nn \__enumext_add_pre_parsep:
1115    {
1116      \int_case:nn { \l__enumext_level_int }
1117        {
1118          { 2 }{
1119                \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1120                  {
1121                    \skip_add:Nn \l__enumext_multicols_above_ii_skip
1122                      {
1123                        \l__enumext_parsep_i_skip
1124                      }
1125                  }
1126              }
1127          { 3 }{
1128                \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1129                  {
1130                    \skip_add:Nn \l__enumext_multicols_above_iii_skip
1131                      {
1132                        \l__enumext_parsep_ii_skip
1133                      }
1134                  }
1135              }
1136          { 4 }{
1137                \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1138                  {
1139                    \skip_add:Nn \l__enumext_multicols_above_iv_skip
1140                      {
1141                        \l__enumext_parsep_iii_skip
1142                      }
1143                  }
1144              }
1145        }
1146    }
```

(*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*⟩.

```
1147  \cs_new_protected:Nn \__enumext_multi_addvspace:
1148    {
1149      \__enumext_multi_set_vskip:
1150      \mode_if_vertical:T
1151        {
1152          \skip_add:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1153            {
1154              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1155            }
1156          \skip_add:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1157            {
1158              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1159            }
1160        }
1161      \__enumext_unskip_unkern: % revisar
1162      \par\nopagebreak
1163      \addvspace{ \skip_use:c { l__enumext_multicols_above_ \__enumext_level: _skip } }
1164    }
```

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

12.21.2   Adjustment of vertical spaces for `multicols` in `keyans`

__enumext_keyans_multi_set_vskip:
__enumext_keyans_multi_addvspace:

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

```
1165 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1166   {
1167     \skip_set:Nn \l__enumext_multicols_above_v_skip
1168       {
1169         \l__enumext_topsep_v_skip
1170       }
1171     \skip_set:Nn \l__enumext_multicols_below_v_skip
1172       {
1173         \l__enumext_topsep_v_skip
1174       }
1175   }
1176 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1177   {
1178     \__enumext_keyans_multi_set_vskip:
1179     \mode_if_vertical:T
1180       {
1181         \skip_add:Nn \l__enumext_multicols_above_v_skip
1182           {
1183             \skip_use:N \l__enumext_partopsep_v_skip
1184           }
1185         \skip_add:Nn \l__enumext_multicols_below_v_skip
1186           {
1187             \skip_use:N \l__enumext_partopsep_v_skip
1188           }
1189       }
1190     \__enumext_unskip_unkern:
1191     \par\nopagebreak
1192     \addvspace{ \l__enumext_multicols_above_v_skip }
1193   }
```

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

12.22   Adjustment of vertical spaces for `minipage`

When nesting a *"list environment"* within the minipage environment, the values of the *"vertical spaces"* are lost. Graphically it can be seen like in the figure 8.
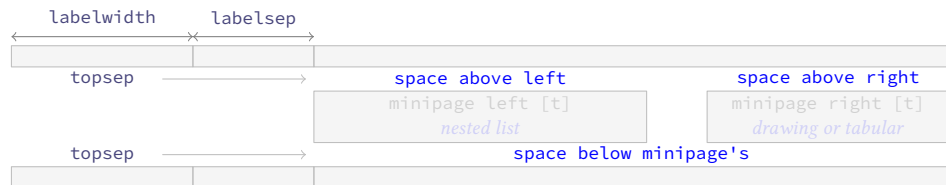
Figure 8: Representation of the minipage spacing adjustment for a nested level.

Since we want to keep the *"left"* and *"right"* environments *"aligned on top"*, preserving the \baselineskip and keep the desired *"spaces"* (\topsep + [\partopsep]) it is necessary to *"adjust"* the *"vertical spaces"* for minipage environments.

Here there are several complications that we must circumvent, the minipage environment eliminates the "top" spaces, the multicols environment can be nested in the minipage environment, the "top" and "bottom" spaces are affected when topsep=0pt and to this is added the \partopsep parameter that comes into action according to whether TeX is in ⟨*horizontal mode*⟩ or ⟨*vertical mode*⟩. Depending on these cases, small adjustments must be made using \vspace and \addvspace to obtain the *"desired vertical spacing"*.

🖋 Again I must make clear that the implementation here is a *"bit questionable"*, but hunting the spaces (glue) produced by the minipage environment is quite complicated, even more if multicols it is nested. The setting of the values was more *"trial and error"* (aprox to \strutbox), using the help of the lua-visual-debug[14] package, again my attempts to find the correct values using \showoutput and \showboxdepth absolutely failed.

12.22.1   Adjustment of vertical spaces for `minipage` in `enumext`

__enumext_minipage_set_skip:
__enumext_minipage_add_space:

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

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

```
1194 \cs_new_protected:Nn \__enumext_minipage_set_skip:
1195   {
```

```
1196      \skip_set:Nn \l__enumext_minipage_right_skip
1197        {
1198          \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1199        }
1200      \mode_if_vertical:T
1201        {
1202          \skip_add:Nn \l__enumext_minipage_right_skip
1203            {
1204              \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1205            }
1206        }
1207      \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:.

```
1208      \skip_set_eq:cN
1209        { l__enumext_multicols_above_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1210      \skip_set_eq:cN
1211        { l__enumext_multicols_below_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1212      \__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.

```
1213      \int_compare:nNnT
1214        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1215        {
1216          \skip_zero:N \topskip
1217          \skip_set_eq:Nc \multicolsep { l__enumext_multicols_above_ \__enumext_level: _skip }
1218        }
1219    }
```

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*.

```
1220  \cs_new_protected:Nn \__enumext_minipage_add_space:
1221    {
1222      \__enumext_minipage_set_skip:
1223      \__enumext_unskip_unkern:
1224      \mode_if_vertical:TF
1225        {
1226          \nopagebreak\nointerlineskip
1227        }
1228        {
1229          \par\nopagebreak\nointerlineskip
1230          \skip_zero:c { l__enumext_partopsep_ \__enumext_level: _skip }
1231        }
1232      \int_compare:nNnTF
1233        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1234        {
1235          \addvspace{ 0.445\box_ht:N \strutbox }
1236        }
1237        {
1238          \addvspace{ 0.250\box_ht:N \strutbox }
1239        }
1240    }
```

(*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.

```
1241  \cs_new_protected:Nn \__enumext_pre_itemsep_skip:
1242    {
1243      \int_case:nn { \l__enumext_level_int }
1244        {
1245          { 2 }{
1246              \skip_if_eq:nnTF
1247                { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1248                {
1249                  \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
```

```
1250                            \skip_set:Nn \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1251                          }
1252                        {
1253                          \dim_compare:nNnT
1254                            { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1255                            {
1256                              \skip_sub:Nn
1257                                \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1258                              \skip_sub:Nn
1259                                \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1260                              \skip_add:Nn
1261                                \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1262                              \skip_add:Nn
1263                                \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1264                            }
1265                          \dim_compare:nNnT
1266                            { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1267                            {
1268                              \skip_set:Nn \l__enumext_minipage_temp_skip
1269                                {
1270                                  \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1271                                }
1272                              \skip_sub:Nn
1273                                \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1274                              \skip_sub:Nn
1275                                \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1276                              \skip_add:Nn
1277                                \l__enumext_minipage_after_skip
1278                                { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1279                              \skip_add:Nn
1280                                \l__enumext_multicols_below_ii_skip
1281                                { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1282                            }
1283                        }
1284                    }
1285            { 3 }{
1286                    \skip_if_eq:nnTF
1287                        { \l__enumext_itemsep_ii_skip } { \c_zero_skip }
1288                        {
1289                          \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1290                          \skip_set:Nn \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1291                        }
1292                        {
1293                          \dim_compare:nNnT
1294                            { \l__enumext_itemsep_ii_skip } < { \l__enumext_minipage_after_skip }
1295                            {
1296                              \skip_sub:Nn
1297                                \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1298                              \skip_sub:Nn
1299                                \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1300                              \skip_add:Nn
1301                                \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1302                              \skip_add:Nn
1303                                \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1304                            }
1305                          \dim_compare:nNnT
1306                            { \l__enumext_itemsep_ii_skip } > { \l__enumext_minipage_after_skip }
1307                            {
1308                              \skip_set:Nn \l__enumext_minipage_temp_skip
1309                                {
1310                                  \l__enumext_itemsep_ii_skip - \l__enumext_minipage_after_skip
1311                                }
1312                              \skip_sub:Nn
1313                                \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1314                              \skip_sub:Nn
1315                                \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1316                              \skip_add:Nn
1317                                \l__enumext_minipage_after_skip
1318                                { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1319                              \skip_add:Nn
1320                                \l__enumext_multicols_below_iii_skip
```

```
1321                                    { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1322                                }
1323                            }
1324                        }
1325            { 4 }{
1326                    \skip_if_eq:nnTF { \l__enumext_itemsep_iii_skip } { \c_zero_skip }
1327                        {
1328                          \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1329                          \skip_set:Nn \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1330                        }
1331                        {
1332                          \dim_compare:nNnT
1333                            { \l__enumext_itemsep_iii_skip } < { \l__enumext_minipage_after_skip }
1334                            {
1335                              \skip_sub:Nn
1336                                \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1337                              \skip_sub:Nn
1338                                \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1339                              \skip_add:Nn
1340                                \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1341                              \skip_add:Nn
1342                                \l__enumext_multicols_below_iv_skip { 0.350\box_ht:N \strutbox }
1343                            }
1344                          \dim_compare:nNnT
1345                            { \l__enumext_itemsep_iii_skip } > { \l__enumext_minipage_after_skip }
1346                            {
1347                              \skip_set:Nn \l__enumext_minipage_temp_skip
1348                                {
1349                                  \l__enumext_itemsep_iii_skip - \l__enumext_minipage_after_skip
1350                                }
1351                              \skip_sub:Nn
1352                                \l__enumext_minipage_after_skip { \l__enumext_itemsep_iii_skip }
1353                              \skip_sub:Nn
1354                                \l__enumext_multicols_below_iv_skip { \l__enumext_itemsep_iii_skip }
1355                              \skip_add:Nn
1356                                \l__enumext_minipage_after_skip
1357                                { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1358                              \skip_add:Nn
1359                                \l__enumext_multicols_below_iv_skip
1360                                { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1361                            }
1362                        }
1363                    }
1364                }
1365        }
```

(*End of definition for* __enumext_pre_itemsep_skip:.)

12.22.2   Adjustment of vertical spaces for `minipage` in keyans

__enumext_keyans_minipage_set_skip:
__enumext_keyans_minipage_add_space:
__enumext_keyans_pre_itemsep_skip:

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

```
1366  \cs_new_protected:Nn \__enumext_keyans_minipage_set_skip:
1367    {
1368      \skip_zero:N \l__enumext_minipage_after_skip
1369      \skip_zero:N \l__enumext_minipage_left_skip
1370      \skip_zero:N \l__enumext_minipage_right_skip
1371      \skip_set:Nn \l__enumext_minipage_right_skip
1372        {
1373          \l__enumext_topsep_v_skip
1374        }
1375      \mode_if_vertical:T
1376        {
1377          \skip_add:Nn \l__enumext_minipage_right_skip
1378            {
1379              \l__enumext_partopsep_v_skip
1380            }
1381        }
1382      \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip
1383      \skip_set_eq:NN \l__enumext_multicols_above_v_skip \l__enumext_minipage_right_skip
1384      \skip_set_eq:NN \l__enumext_multicols_below_v_skip \l__enumext_minipage_right_skip
```

```
1385        \__enumext_keyans_pre_itemsep_skip:
1386        \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
1387          {
1388            \skip_zero:N \topskip
1389            \skip_set_eq:NN \multicolsep \l__enumext_minipage_right_skip
1390          }
1391      }
1392  \cs_new_protected:Nn \__enumext_keyans_minipage_add_space:
1393    {
1394      \__enumext_keyans_minipage_set_skip:
1395      \__enumext_unskip_unkern:
1396      \mode_if_vertical:TF
1397        {
1398          \nopagebreak\nointerlineskip
1399        }
1400        {
1401          \par\nopagebreak\nointerlineskip
1402          \skip_zero:N \l__enumext_partopsep_v_skip
1403        }
1404      \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1405        {
1406          \addvspace{ 0.445\box_ht:N \strutbox }
1407        }
1408        {
1409          \addvspace{ 0.250\box_ht:N \strutbox }
1410        }
1411    }
1412  \cs_new_protected:Nn \__enumext_keyans_pre_itemsep_skip:
1413    {
1414      \skip_if_eq:nnTF
1415        { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1416        {
1417          \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1418          \skip_set:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1419        }
1420        {
1421          \dim_compare:nNnT
1422            { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1423            {
1424              \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1425              \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1426              \skip_add:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1427              \skip_add:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1428            }
1429          \dim_compare:nNnT
1430            { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1431            {
1432              \skip_set:Nn \l__enumext_minipage_temp_skip
1433                {
1434                  \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1435                }
1436              \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1437              \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1438              \skip_add:Nn \l__enumext_minipage_after_skip
1439                { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1440              \skip_add:Nn \l__enumext_multicols_below_v_skip
1441                { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1442            }
1443        }
1444    }
```

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

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

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

```
1445  \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1446    {
1447      \skip_zero_new:N \l__enumext_minipage_left_skip
```

```
1448      \skip_gzero_new:N \g__enumext_minipage_right_skip
1449      \skip_gzero_new:N \g__enumext_minipage_after_skip
1450      \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1451        {
1452          \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1453          \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1454        }
1455        {
1456          \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1457          \skip_gset:Nn \g__enumext_minipage_right_skip
1458            {
1459              \l__enumext_topsep_vii_skip
1460            }
1461          \skip_gset:Nn \g__enumext_minipage_after_skip
1462            {
1463              0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1464            }
1465        }
1466    }
1467  \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1468    {
1469      \skip_zero_new:N \l__enumext_minipage_after_skip
1470      \skip_zero_new:N \l__enumext_minipage_left_skip
1471      \skip_zero_new:N \l__enumext_minipage_right_skip
1472      \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1473        {
1474          \skip_set:Nn \l__enumext_minipage_left_skip
1475            {
1476              0.5\box_dp:N \strutbox
1477            }
1478          \skip_set:Nn \l__enumext_minipage_right_skip
1479            {
1480              \l__enumext_partopsep_viii_skip
1481            }
1482          \skip_set:Nn \l__enumext_minipage_after_skip
1483            {
1484              1.6\box_dp:N \strutbox
1485            }
1486        }
1487        {
1488          \skip_set:Nn \l__enumext_minipage_left_skip
1489            {
1490              0.5875\box_dp:N \strutbox
1491            }
1492          \skip_set:Nn \l__enumext_minipage_right_skip
1493            {
1494              \l__enumext_topsep_viii_skip
1495            }
1496          \skip_set:Nn \l__enumext_minipage_after_skip
1497            {
1498              0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1499            }
1500        }
1501    }
```

(*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.

```
1502  \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1503    {
1504      \__enumext_mini_set_vskip_vii:
1505      \par\nopagebreak
1506      \addvspace { \l__enumext_minipage_left_skip }
1507    }
1508  \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1509    {
1510      \__enumext_mini_set_vskip_viii:
```

```
1511      \par\nopagebreak
1512      \addvspace { \l__enumext_minipage_left_skip }
1513    }
```

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

12.22.4 The command \miniright

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

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

```
1514  \NewDocumentCommand \miniright { s }
1515    {
1516      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1517        {
1518          \msg_error:nnn { enumext } { wrong-miniright-place }
1519        }
1520      % outside
1521      \bool_lazy_and:nnT
1522        { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
1523        { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
1524        {
1525          \msg_error:nnn { enumext } { wrong-miniright-place }
1526        }
1527      % starred env
1528      \bool_if:NT \l__enumext_starred_bool
1529        {
1530          \msg_error:nnn { enumext } { wrong-miniright-starred }
1531        }
1532      \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
1533        {
1534          \__enumext_keyans_mini_right_cmd:n {#1}
1535        }
1536        { \__enumext_mini_right_cmd:n {#1} }
1537    }
```

(*End of definition for* \miniright. *This function is documented on page 10.*)

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

```
1538  \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1539    {
1540      \dim_compare:nNnTF
1541        { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
1542        {
1543          \__enumext_multicols_stop:
1544          \int_compare:nNnT
1545            { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } = { 1 }
1546            {
1547              \par\addvspace{ \l__enumext_minipage_after_skip }
1548            }
1549          \end__enumext_mini_page
1550          \hfill
1551          \__enumext_mini_page{ \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } }
1552            \par\nointerlineskip
1553            \addvspace { \l__enumext_minipage_right_skip }
1554            \bool_if:nF {#1}
1555              {
1556                \centering
1557              }
1558            \int_gzero:N \g__enumext_minipage_stat_int
```

```
1559            }
1560          { \msg_error:nnn { enumext } { wrong-miniright-use } }
1561      % paranoia
1562      \RenewDocumentCommand \miniright { s }
1563        {
1564          \msg_error:nn { enumext } { many-miniright-used }
1565        }
1566    }
```

(*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.

```
1567 \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1568    {
1569      \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1570        {
1571          \__enumext_keyans_multicols_stop:
1572          \int_compare:nNnT { \l__enumext_columns_v_int } = { 1 }
1573            {
1574              \par\addvspace{ \l__enumext_minipage_after_skip }
1575            }
1576          \end__enumext_mini_page
1577          \hfill
1578          \__enumext_mini_page{ \l__enumext_minipage_right_v_dim }
1579            \par\nointerlineskip
1580            \addvspace { \l__enumext_minipage_right_skip }
1581            \bool_if:nF {#1}
1582              {
1583                \centering
1584              }
1585            \int_gzero:N \g__enumext_minipage_stat_int
1586          }
1587        { \msg_error:nnn { enumext } { wrong-miniright-use } }
1588      % paranoia
1589      \RenewDocumentCommand \miniright { s }
1590        {
1591          \msg_error:nn { enumext } { many-miniright-used }
1592        }
1593    }
```

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

12.23   Setting above and below keys

While having controlled the *vertical spaces* within the enumext and keyans environments when using the columns or mini-env keys, sometimes the *"vertical spaces above"* or *"vertical spaces below"* the environments are not as expected and it is necessary to be able to apply a *"fine correction"* to these. As I have not been able to correct these *glitches*, the best option is to leave a couple of ⟨*keys*⟩ dedicated to this purpose, in this case it is best to use \vspace or \vspace* when convenient.

above   Define above, above*, below and below* keys for enumext and keyans environments.
above*
below
below*
```
1594 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1595    {
1596      \keys_define:nn { enumext / #1 }
1597        {
1598          above  .skip_set:c = { l__enumext_vspace_above_#2_skip },
1599          above  .value_required:n = true,
1600          above* .code:n     = \bool_set_true:c { l__enumext_vspace_a_star_#2_bool }
1601                               \keys_set:nn { enumext / #1 } { above = {##1} },
1602          above* .value_required:n = true,
1603          below  .skip_set:c = { l__enumext_vspace_below_#2_skip },
1604          below  .value_required:n = true,
1605          below* .code:n     = \bool_set_true:c { l__enumext_vspace_b_star_#2_bool }
1606                               \keys_set:nn { enumext / #1 } { below = {##1} },
1607          below* .value_required:n = true,
1608        }
1609    }
1610 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

12.23.1 Functions for above and below keys in enumext

__enumext_vspace_above:

The function __enumext_vspace_above: apply the *vertical space above* the enumext environment set by the above* and above keys.

```
1611  \cs_new_protected:Nn \__enumext_vspace_above:
1612    {
1613      \skip_if_eq:nnF
1614        { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1615        {
1616          \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
1617            {
1618              \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1619            }
1620            {
1621              \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1622            }
1623        }
1624    }
```

(*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.

```
1625  \cs_new_protected:Nn \__enumext_vspace_below:
1626    {
1627      \skip_if_eq:nnF
1628        { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1629        {
1630          \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1631            {
1632              \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1633            }
1634            {
1635              \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1636            }
1637        }
1638    }
```

(*End of definition for* __enumext_vspace_below:.)

12.23.2 Functions for above and below keys in keyans

__enumext_vspace_above_v:

The function __enumext_vspace_above_v: apply the *vertical space above* the keyans environment set by the above and above* keys.

```
1639  \cs_new_protected:Nn \__enumext_vspace_above_v:
1640    {
1641      \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1642        {
1643          \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1644            {
1645              \vspace*{ \l__enumext_vspace_above_v_skip }
1646            }
1647            { \vspace { \l__enumext_vspace_above_v_skip } }
1648        }
1649    }
```

(*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.

```
1650  \cs_new_protected:Nn \__enumext_vspace_below_v:
1651    {
1652      \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1653        {
1654          \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1655            {
1656              \vspace*{ \l__enumext_vspace_below_v_skip }
1657            }
1658            { \vspace { \l__enumext_vspace_below_v_skip } }
1659        }
1660    }
```

(*End of definition for* __enumext_vspace_below_v:.)

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

__enumext_vspace_above_vii:
__enumext_vspace_above_viii:

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

```
1661 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1662   {
1663     \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1664       {
1665         \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1666           {
1667             \vspace*{ \l__enumext_vspace_above_vii_skip }
1668           }
1669           { \vspace { \l__enumext_vspace_above_vii_skip } }
1670       }
1671   }
1672 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1673   {
1674     \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1675       {
1676         \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1677           {
1678             \vspace*{ \l__enumext_vspace_above_viii_skip }
1679           }
1680           { \vspace { \l__enumext_vspace_above_viii_skip } }
1681       }
1682   }
```

(*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.

```
1683 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1684   {
1685     \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1686       {
1687         \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1688           {
1689             \vspace*{ \l__enumext_vspace_below_vii_skip }
1690           }
1691           { \vspace { \l__enumext_vspace_below_vii_skip } }
1692       }
1693   }
1694 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1695   {
1696     \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1697       {
1698         \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1699           {
1700             \vspace*{ \l__enumext_vspace_below_viii_skip }
1701           }
1702           { \vspace { \l__enumext_vspace_below_viii_skip } }
1703       }
1704   }
```

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

12.24 Setting series, resume and resume* keys

The series key is responsible for the whole process of the resume and resume* keys. The idea behind this is to be able to absorb the ⟨*keys*⟩ passed to the *optional argument* of the *"first level"* of the environments enumext and enumext*, but, discarding some specific ⟨*keys*⟩. This implementation is adapted directly from the code provided by Jonathan P. Spratte (@Skillmon) in chat-TeX-SX

series
resume
resume*

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

```
1705 \cs_set_protected:Npn \__enumext_tmp:n #1
1706   {
1707     \keys_define:nn { enumext / #1 }
1708       {
1709         series  .str_set:N = \l__enumext_series_str,
1710         series  .value_required:n = true,
1711         resume  .code:n = \__enumext_resume_series:n {##1},
1712         resume* .code:n = \__enumext_resume_starred:,
```

```
1713        resume* .value_forbidden:n = true,
1714      }
1715    }
1716  \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

12.24.1   Internal functions for series key

__enumext_filter_series:n

__enumext_filter_series_key:n

__enumext_filter_series_pair:nn

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

```
1717  \cs_new:Npn \__enumext_filter_series:n #1
1718    {
1719      \use:e
1720        {
1721          \keyval_parse:NNn
1722            \__enumext_filter_series_key:n
1723            \__enumext_filter_series_pair:nn {#1}
1724        }
1725    }
```

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.

```
1726  \cs_new:Npn \__enumext_filter_series_key:n #1
1727    {
1728      \str_case:nnF {#1}
1729        {
1730          { resume } {} { resume* } {} { base-fix } {}
1731        }
1732        { , { \exp_not:n {#1} } }
1733    }
```

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.

```
1734  \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1735    {
1736      \str_case:nnF {#1}
1737        {
1738          { series } {} { resume } {} { start } {}
1739          { start* } {}  { save-ans } {} { save-key } {}
1740        }
1741        { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1742    }
```

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

__enumext_parse_series:n

__enumext_resume_last:n

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

🎯 The value of boolean variable \l__enumext_resume_active_bool is set to true by the function __enumext_resume_- counter:n which is used by the keys resume and resume*, in this case we must Make sure it is set to false so that it does not overwrite the default filtered ⟨*keys*⟩. This function is passed to the function __enumext_parse_keys:n in the enumext environment definition (§12.38) and to the function __enumext_parse_keys_vii:n in the enumext* environment definition (§12.43).

```
1743  \cs_new_protected:Npn \__enumext_parse_series:n #1
1744    {
1745      \str_if_empty:NTF \l__enumext_series_str
1746        {
1747          \bool_if:NF \l__enumext_resume_active_bool
1748            {
1749              \__enumext_resume_last:n {#1}
1750            }
1751        }
1752        {
1753          \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str _tl }
1754          \tl_gset:ce { g__enumext_series_ \l__enumext_series_str _tl }
1755            { \__enumext_filter_series:n {#1} }
1756          \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str _int }
1757            {
```

```
1758              \int_new:c { g__enumext_series_ \l__enumext_series_str _int }
1759            }
1760          }
1761      }
```

The function `\__enumext_resume_last:n` will be in charge of saving the filtering ⟨*keys*⟩ when the `series` key is *not used* and will save them in the variable `\g__enumext_standar_series_tl` for the `enumext` environment and in the variable `\g__enumext_starred_series_tl` for the `enumext*` environment. Here we must use `\bool_lazy_all:nT` to make sure that the default values are not overwritten when the environment is nested and the `series` key is not being used.

```
1762  \cs_new_protected:Npn \__enumext_resume_last:n #1
1763    {
1764      \bool_if:NT \l__enumext_standar_first_bool
1765        {
1766          \tl_gclear:N \g__enumext_standar_series_tl
1767          \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1768        }
1769      \bool_if:NT \l__enumext_starred_first_bool
1770        {
1771          \tl_gclear:N \g__enumext_starred_series_tl
1772          \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1773        }
1774    }
```

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

12.24.2 Internal function to save counter value

`\__enumext_resume_save_counter:` The `\__enumext_resume_save_counter:` function will save the last counter value to `\g__enumext_-series_`⟨*series name*⟩`_int` if the `series={`⟨*series name*⟩`}` key has been passed, to `\g__enumext_resume_-int` if it has passed the key `resume` *without value* and the key `series` is not active, in `\g__enumext_series_-`⟨*series name*⟩`_int` if the key `resume={`⟨*series name*⟩`}` has been passed and in `\g__enumext_series_`⟨*store name*⟩`_int` if the key has been passed `save-ans={`⟨*store name*⟩`}`.

🌶 The variables `\l__enumext_series_str` and `\l__enumext__resume_name_tl` contain the same `{`⟨*series name*⟩`}` but are executed at different moments, the integer variable with `\l__enumext_series_str` sets the value when execute `series={`⟨*series name*⟩`}` and the integer variable with `\l__enumext__resume_name_tl` sets the subsequent values when use `resume={`⟨*series name*⟩`}`. This function is passed to the `enumext` environment definition (§12.38) and the `enumext*` environment definition (§12.43).

```
1775  \cs_new_protected:Nn \__enumext_resume_save_counter:
1776    {
1777      \bool_if:NT \g__enumext_standar_bool
1778        {
1779          \tl_if_empty:NF \l__enumext_series_str
1780            {
1781              \int_gset_eq:cN
1782                { g__enumext_series_ \l__enumext_series_str _int } \value{enumXi}
1783            }
1784          \tl_if_empty:NTF \l__enumext_resume_name_tl
1785            {
1786              \str_if_empty:NT \l__enumext_series_str
1787                {
1788                  \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1789                }
1790            }
1791            {
1792              \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1793                {
1794                  \int_gset_eq:cN
1795                    { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXi}
1796                }
1797            }
1798          \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1799            {
1800              \int_gset_eq:cN
1801                { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1802            }
1803        }
1804      \bool_if:NT \g__enumext_starred_bool
1805        {
1806          \tl_if_empty:NF \l__enumext_series_str
1807            {
```

```
1808              \int_gset_eq:cN
1809                { g__enumext_series_ \l__enumext_series_str _int } \value{enumXvii}
1810            }
1811          \tl_if_empty:NTF \l__enumext_resume_name_tl
1812            {
1813              \str_if_empty:NT \l__enumext_series_str
1814                {
1815                  \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1816                }
1817            }
1818            {
1819              \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1820                {
1821                  \int_gset_eq:cN
1822                    { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXvii}
1823                }
1824            }
1825          \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1826            {
1827              \int_gset_eq:cN
1828                { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1829            }
1830        }
1831    }
```

(*End of definition for* __enumext_resume_save_counter:.)

12.24.3 Internal functions for resume key

__enumext_resume_series:n   The function __enumext_resume_series:n will handle the argument passed to the resume key in enumext and enumext* environments. If the key is passed *without value* the function __enumext_resume_counter: is executed which will set the counter according to the numbering of the last enumext or enumext* environments in which series={⟨*series name*⟩} key is not present, if the save-ans key is active it will set the counter according to the value of the integer variable created by that key, otherwise it will verify that the \g__enumext_series_⟨*series name*⟩_tl variable set by the series key exists, if so it will pass these keys to the *first level* of the environment, otherwise it will return an error.

```
1832  \cs_new_protected:Npn \__enumext_resume_series:n #1
1833    {
1834      \tl_if_empty:nTF {#1}
1835        {
1836          \__enumext_resume_counter:n { }
1837        }
1838        {
1839          \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1840            {
1841              \__enumext_resume_counter:n {#1}
1842              \bool_if:NT \g__enumext_standar_bool
1843                {
1844                  \keys_set:nv { enumext / level-1 }
1845                    { g__enumext_series_ \tl_to_str:n {#1} _tl }
1846                }
1847              \bool_if:NT \g__enumext_starred_bool
1848                {
1849                  \keys_set:nv { enumext / enumext* }
1850                    { g__enumext_series_ \tl_to_str:n {#1} _tl }
1851                }
1852            }
1853            {
1854              \bool_if:NT \g__enumext_standar_bool
1855                {
1856                  \msg_error:nnn { enumext } { unknown-series } {#1}
1857                }
1858              \bool_if:NT \g__enumext_starred_bool
1859                {
1860                  \msg_error:nnn { enumext } { unknown-series } {#1}
1861                }
1862            }
1863        }
1864    }
```

(*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`.

```
1865 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1866   {
1867     \bool_set_true:N \l__enumext_resume_active_bool
1868     \tl_set:Nn \l__enumext_resume_name_tl {#1}
1869     \tl_if_empty:NTF \l__enumext_resume_name_tl
1870       {
1871         \__enumext_resume_counter:
1872       }
1873       {
1874         \__enumext_resume_counter_series:
1875       }
1876     \__enumext_resume_counter_save_ans:
1877   }
```

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.

```
1878 \cs_new_protected:Nn \__enumext_resume_counter:
1879   {
1880     \bool_if:NT \g__enumext_standar_bool
1881       {
1882         \int_gincr:N \g__enumext_resume_int
1883         \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
1884       }
1885     \bool_if:NT \g__enumext_starred_bool
1886       {
1887         \int_gincr:N \g__enumext_resume_vii_int
1888         \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
1889       }
1890   }
```

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.

```
1891 \cs_new_protected:Nn \__enumext_resume_counter_series:
1892   {
1893     \bool_if:NT \g__enumext_standar_bool
1894       {
1895         \int_set:Nn \l__enumext_start_i_int
1896           {
1897             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1898           }
1899       }
1900     \bool_if:NT \g__enumext_starred_bool
1901       {
1902         \int_set:Nn \l__enumext_start_vii_int
1903           {
1904             \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
1905           }
1906       }
1907   }
```

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.

```
1908 \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
1909   {
1910     \bool_lazy_and:nnT
1911       { \bool_if_p:N \l__enumext_standar_first_bool }
1912       { \bool_if_p:N \l__enumext_store_active_bool }
1913       {
1914         \int_set:Nn \l__enumext_start_i_int
1915           {
1916             \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1917           }
```

```
1918        }
1919    \bool_lazy_and:nnT
1920      { \bool_if_p:N \l__enumext_starred_first_bool }
1921      { \bool_if_p:N \l__enumext_store_active_bool }
1922      {
1923        \int_set:Nn \l__enumext_start_vii_int
1924          {
1925            \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
1926          }
1927      }
1928    }
```

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

12.24.4   Internal function for `resume*` key

`\__enumext_resume_starred:` The function `\__enumext_resume_starred:` will handle the `resume*` key in the enumext and enumext* environments. This function will execute the filtered ⟨*keys*⟩ in the last one and will continue with the numbering according to the last execution of the environment enumext or enumext* in which the keys `resume={`⟨*series name*⟩`}` or `series={`⟨*series name*⟩`}` were not active.

```
1929 \cs_new_protected:Nn \__enumext_resume_starred:
1930    {
1931      \bool_if:NT \g__enumext_standar_bool
1932        {
1933          \tl_if_empty:NF \g__enumext_standar_series_tl
1934            {
1935              \__enumext_resume_counter:n { }
1936              \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
1937            }
1938        }
1939      \bool_if:NT \g__enumext_starred_bool
1940        {
1941          \tl_if_empty:NF \g__enumext_starred_series_tl
1942            {
1943              \__enumext_resume_counter:n { }
1944              \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
1945            }
1946        }
1947    }
```

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

12.25   Setting `save-ans`, `check-ans` and `no-store` keys

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

12.25.1   Setting `save-ans` key

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

```
1948 \cs_set_protected:Npn \__enumext_tmp:n #1
1949    {
1950      \keys_define:nn { enumext / #1 }
1951        {
1952          save-ans .code:n = \__enumext_storing_set:n {##1},
1953          save-ans .value_required:n = true,
1954        }
1955    }
1956 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

12.25.2   Internal functions for `save-ans` key

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

```
1957 \cs_new_protected:Nn \__enumext_start_save_ans_msg:
1958    {
1959      \msg_term:nnVV { enumext } { save-ans-log }
1960        \g__enumext_envir_name_tl \l__enumext_store_name_tl
1961    }
```

```
1962  \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
1963    {
1964      \msg_term:nnVV { enumext } { save-ans-log-hook }
1965        \g__enumext_envir_name_tl \g__enumext_store_name_tl
1966    }
```

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

__enumext_storing_set:n
__enumext_storing_exec:

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

```
1967  \cs_new_protected:Npn \__enumext_storing_set:n #1
1968    {
1969      \tl_set:Ne \l__enumext_store_name_tl {#1}
1970      \tl_if_empty:NTF \l__enumext_store_name_tl
1971        {
1972          \bool_lazy_or:nnT
1973            { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1974            {
1975              \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
1976            }
1977        }
1978        {
1979          \bool_lazy_or:nnT
1980            { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
1981            {
1982              \__enumext_start_save_ans_msg:
1983              \__enumext_storing_exec:
1984            }
1985        }
1986    }
```

The function __enumext_storing_exec: will set to true the variable \l__enumext_store_active_bool which activates the use of the \anskey command and the keyans, keyans* and keyanspic environments and will set to true the variable \l__enumext_check_answers_bool used for checking answers by the check-ans and no-store keys, copy {⟨*store name*⟩} into the global variable \g__enumext_store_name_-tl and execute the function __enumext_anskey_env_make:V creating the environment anskey* (§12.30). The ⟨*prop list*⟩ \g__enumext_series_⟨*store name*⟩_prop and the ⟨*sequence*⟩ \g__enumext_series_⟨*store name*⟩_seq will be created globally to *"store content"* in case they do not exist together with the integer variable \g__enumext_series_⟨*store name*⟩_int used by the keys resume and resume*.

```
1987  \cs_new_protected:Nn \__enumext_storing_exec:
1988    {
1989      \bool_set_true:N \l__enumext_store_active_bool
1990      \bool_set_true:N \l__enumext_check_answers_bool
1991      \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
1992      \__enumext_anskey_env_make:V \l__enumext_store_name_tl
1993      \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
1994        {
1995          \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
1996          \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
1997        }
1998      \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
1999        {
2000          \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
2001          \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
2002        }
2003      \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
2004        {
2005          \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
2006          \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
2007        }
2008    }
```

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

12.25.3   The check answer mechanism

The mechanism for checking that all questions are answered follows this logic:

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

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

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

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

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

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

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

12.25.4 Setting check-ans and no-store keys

check-ans
no-store

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

```
2009 \cs_set_protected:Npn \__enumext_tmp:n #1
2010   {
2011     \keys_define:nn { enumext / #1 }
2012       {
2013         check-ans .bool_set:N = \l__enumext_check_ans_key_bool,
2014         check-ans .initial:n  = false,
2015         check-ans .value_required:n = true,
2016         no-store  .code:n = {
2017                              \bool_set_false:N \l__enumext_check_answers_bool
2018                              \bool_set_false:N \l__enumext_check_ans_key_bool
2019                            },
2020         no-store  .value_forbidden:n = true,
2021       }
2022   }
2023 \clist_map_inline:nn
2024   {
2025     level-1, level-2, level-3, level-4, enumext*
2026   }
2027   { \__enumext_tmp:n {#1} }
```

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

12.25.5 Set-up check answer mechanism

__enumext_check_ans_active:
__enumext_check_ans_level:

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

```
2028 \cs_new_protected:Nn \__enumext_check_ans_active:
2029   {
2030     \tl_if_empty:NF \l__enumext_store_name_tl
2031       {
2032         \bool_if:NT \l__enumext_check_answers_bool
2033           {
2034             \__enumext_check_ans_level:
2035           }
2036       }
2037   }
```

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"*.

```
2038  \cs_new_protected:Nn \__enumext_check_ans_level:
2039    {
2040      \int_case:nn { \l__enumext_level_int }
2041        {
2042          { 1 }{
2043                \bool_lazy_all:nT
2044                  {
2045                    { \bool_if_p:N \g__enumext_starred_bool }
2046                    { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
2047                  }
2048                  {
2049                    \int_gdecr:N \g__enumext_item_number_int
2050                    \bool_set_false:N \l__enumext_item_number_bool
2051                  }
2052              }
2053          { 2 }{
2054                \int_gdecr:N \g__enumext_item_number_int
2055                \bool_set_false:N \l__enumext_item_number_bool
2056              }
2057          { 3 }{
2058                \int_gdecr:N \g__enumext_item_number_int
2059                \bool_set_false:N \l__enumext_item_number_bool
2060              }
2061          { 4 }{
2062                \int_gdecr:N \g__enumext_item_number_int
2063                \bool_set_false:N \l__enumext_item_number_bool
2064              }
2065        }
```

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.

```
2066      \int_case:nn { \l__enumext_level_h_int }
2067        {
2068          { 1 }{
2069                \bool_lazy_all:nT
2070                  {
2071                    { \bool_if_p:N \g__enumext_standar_bool }
2072                    { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
2073                  }
2074                  {
2075                    \int_gdecr:N \g__enumext_item_number_int
2076                    \bool_set_false:N \l__enumext_item_number_bool
2077                  }
2078              }
2079        }
2080    }
```

(*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.

```
2081  \cs_new_protected:Nn \__enumext_check_ans_key_hook:
2082    {
2083      \bool_lazy_and:nnT
2084        { \bool_if_p:N \l__enumext_check_ans_key_bool }
2085        { \bool_if_p:N \g__enumext_standar_bool }
2086        {
2087          \bool_gset_true:N \g__enumext_check_ans_key_bool
2088        }
2089      \bool_lazy_and:nnT
2090        { \bool_if_p:N \l__enumext_check_ans_key_bool }
2091        { \bool_if_p:N \g__enumext_starred_bool }
2092        {
2093          \bool_gset_true:N \g__enumext_check_ans_key_bool
2094        }
2095    }
```

(*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:.

```
2096 \cs_new_protected:Nn \__enumext_item_answer_diff:
2097   {
2098     \int_gset:Nn \g__enumext_item_answer_diff_int
2099       {
2100         \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
2101       }
2102   }
```

(*End of definition for* __enumext_item_answer_diff:.)

__enumext_check_ans_show:
__enumext_check_ans_msg_less:
__enumext_check_ans_msg_same_ok:
__enumext_check_ans_msg_greater:

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

```
2103 \cs_new_protected:Nn \__enumext_check_ans_show:
2104   {
2105     \int_case:nn { \g__enumext_item_answer_diff_int }
2106       {
2107         { -1 }{ \__enumext_check_ans_msg_less:      }
2108         {  0 }{ \__enumext_check_ans_msg_same_ok: }
2109         {  1 }{ \__enumext_check_ans_msg_greater: }
2110       }
2111   }
2112 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
2113   {
2114     \msg_warning:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2115       { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2116   }
2117 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
2118   {
2119     \msg_term:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2120       { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2121   }
2122 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
2123   {
2124     \msg_warning:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2125       { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2126   }
```

(*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:.

```
2127 \cs_new_protected:Nn \__enumext_check_ans_log:
2128   {
2129     \int_case:nn { \g__enumext_item_answer_diff_int }
2130       {
2131         { -1 }{ \__enumext_check_ans_log_msg_less:      }
2132         {  0 }{ \__enumext_check_ans_log_msg_same_ok: }
2133         {  1 }{ \__enumext_check_ans_log_msg_greater: }
2134       }
2135   }
2136 \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
2137   {
2138     \msg_log:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2139       { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2140   }
2141 \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
2142   {
2143     \msg_log:nneee { enumext } { items-same-answer }  { \g__enumext_store_name_tl }
2144       { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2145   }
```

```
2146  \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
2147    {
2148      \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2149        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2150    }
```

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

12.25.6 Check for \item* and \anspic* commands

__enumext_check_starred_cmd:n   The function __enumext_check_starred_cmd:n performs an extra check for the keyans, keyans* and keyanspic environments. Unlike the check executed by check-ans key this one is not controlled by any key, it is intended to prevent the forgetting of \item* or \anspic* in these environments.

```
2151  \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
2152    {
2153      \int_compare:nNnT
2154        { \g__enumext_check_starred_cmd_int } = { 0 }
2155        {
2156          \msg_warning:nnnV
2157            { enumext } { missing-starred }{ #1 } \l__enumext_check_start_line_env_tl
2158        }
2159      \int_compare:nNnT
2160        { \g__enumext_check_starred_cmd_int } > { 1 }
2161        {
2162          \msg_warning:nnnV
2163            { enumext } { many-starred }{ #1 } \l__enumext_check_start_line_env_tl
2164        }
2165      \int_gzero:N \g__enumext_check_starred_cmd_int
2166      \tl_clear:N \l__enumext_check_start_line_env_tl
2167    }
```

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

12.26 Keys and functions associated with storage

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

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

```
2168  \cs_set_protected:Npn \__enumext_tmp:n #1
2169    {
2170      \keys_define:nn { enumext / #1 }
2171        {
2172          wrap-ans    .cs_set_protected:Np = \__enumext_anskey_wrapper:n ##1,
2173          wrap-ans    .initial:n =
2174                        {
2175                          \fbox{\parbox[t]{\dimeval{\itemwidth -2\fboxsep -2\fboxrule}}{##1}}
2176                        },
2177          wrap-ans    .value_required:n = true,
2178          wrap-opt    .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
2179          wrap-opt    .initial:n = [{##1}],
2180          wrap-opt    .value_required:n = true,
2181          save-sep    .tl_set:N  = \l__enumext_store_keyans_item_opt_sep_tl,
2182          save-sep    .initial:n = {, ~ },
2183          save-sep    .value_required:n = true,
2184          mark-ans    .tl_set:N  = \l__enumext_mark_answer_sym_tl,
2185          mark-ans    .initial:n = \textasteriskcentered,
2186          mark-ans    .value_required:n = true,
2187          mark-pos    .choice:,
2188          mark-pos / left     .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2189          mark-pos / right    .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2190          mark-pos / unknown .code:n =
2191                            \msg_error:nneee { enumext } { unknown-choice }
2192                              { mark-pos } { left, ~ right } { \exp_not:n {##1} },
2193          mark-pos    .initial:n = right,
2194          mark-pos    .value_required:n = true,
2195          show-ans    .bool_set:N = \l__enumext_show_answer_bool,
2196          show-ans    .initial:n  = false,
2197          show-ans    .value_required:n = true,
2198          show-pos    .bool_set:N = \l__enumext_show_position_bool,
2199          show-pos    .initial:n  = false,
2200          show-pos    .value_required:n = true,
2201          mark-ref    .tl_set:N   = \l__enumext_mark_ref_sym_tl,
```

```
2202        mark-ref    .initial:n  = \textasteriskcentered,
2203        mark-ref    .value_required:n = true,
2204        save-ref    .bool_set:N = \l__enumext_store_ref_key_bool,
2205        save-ref    .initial:n  = false,
2206        save-ref    .value_required:n = true,
2207      }
2208    }
2209  \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

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

show-ans
```
2210  \cs_set_protected:Npn \__enumext_tmp:n #1
```
show-pos
```
2211    {
2212      \keys_define:nn { enumext / #1 }
2213        {
2214          mark-pos .choice:,
2215          mark-pos / left   .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2216          mark-pos / right  .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2217          mark-pos .initial:n = right,
2218          mark-pos .value_required:n  = true,
2219          show-ans .bool_set:N = \l__enumext_show_answer_bool,
2220          show-ans .initial:n  = false,
2221          show-ans .value_required:n = true,
2222          show-pos .bool_set:N = \l__enumext_show_position_bool,
2223          show-pos .initial:n  = false,
2224          show-pos .value_required:n = true,
2225        }
2226    }
2227  \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }
```

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

12.26.1 Store optional arguments of the environments

The idea behind *"storing"* in the ⟨*sequence*⟩ is to have a copy of the structure of the environment in which the key save-ans is being executed so we must capture the *optional argument* passed to the levels of the environment in which it is executed and *"storing"* them.

__enumext_store_active_keys:n   The functions __enumext_store_active_keys:n and __enumext_store_active_keys_vii:n will

__enumext_store_active_keys_vii:n   be responsible for *"storing"* the ⟨*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*⟩.

```
2228  \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2229    {
2230      \bool_if:cF { l__enumext_store_save_key_ \__enumext_level: _bool }
2231        {
2232          \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2233          \tl_set:ce
2234            { l__enumext_store_save_key_ \__enumext_level: _tl }
2235            { \__enumext_filter_save_key:n {#1} }
2236        }
2237    }
2238  \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2239    {
2240      \bool_if:NF \l__enumext_store_save_key_vii_bool
2241        {
2242          \tl_clear:N \l__enumext_store_save_key_vii_tl
2243          \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2244        }
2245    }
```

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

12.26.2 Setting save-key key

Since this list structure will be stored in the ⟨*sequence*⟩ established by the `save-ans` key when executing `\anskey`, we will not be able to modify it. The best thing here is to have a key that allows you to modify the *optional argument* of the list stored in the ⟨*sequence*⟩.

save-key The values set by this key passed in the *optional 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`.

Define the key `save-key` for all levels of `enumext` and `enumext*` environments.

```
2246 \cs_set_protected:Npn \__enumext_tmp:n #1
2247   {
2248     \keys_define:nn { enumext / enumext* }
2249       {
2250         save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2251         save-key .value_required:n = true,
2252       }
2253     \keys_define:nn { enumext / #1 }
2254       {
2255         save-key .code:n = \__enumext_parse_save_key:n {##1},
2256         save-key .value_required:n = true,
2257       }
2258   }
2259 \clist_map_inline:nn { level-1, level-2, level-3, level-4 } { \__enumext_tmp:n {#1} }
```

(*End of definition for* `save-key`.)

__enumext_parse_save_key:n
__enumext_parse_save_key_vii:n

The functions `\__enumext_parse_save_key:n` and `\__enumext_parse_save_key_vii:n` will be responsible for storing the filtered ⟨*keys*⟩ in the variable `\l__enumext_store_save_key_X_tl` for `enumext` and `enumext*`.

```
2260 \cs_new_protected:Npn \__enumext_parse_save_key:n #1
2261   {
2262     \bool_set_true:c { l__enumext_store_save_key_ \__enumext_level: _bool }
2263     \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2264     \tl_set:ce
2265       { l__enumext_store_save_key_ \__enumext_level: _tl }
2266       { \__enumext_filter_save_key:n {#1} }
2267   }
2268 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
2269   {
2270     \bool_set_true:N \l__enumext_store_save_key_vii_bool
2271     \tl_clear:N \l__enumext_store_save_key_vii_tl
2272     \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2273   }
```

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

12.26.3 Internal functions to store optional arguments

__enumext_filter_save_key:n
__enumext_filter_save_key_key:n
__enumext_filter_save_key_pair:nn

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

```
2274 \cs_new:Npn \__enumext_filter_save_key:n #1
2275   {
2276     \use:e
2277       {
2278         \keyval_parse:NNn
2279           \__enumext_filter_save_key_key:n
2280           \__enumext_filter_save_key_pair:nn {#1}
2281       }
2282   }
```

The function `\__enumext_filter_save_key_key:n` will be responsible for filtering the ⟨*keys*⟩ that are passed "*without value*" by excluding the `resume`, `resume*`, `no-store` and `base-fix` keys.

```
2283 \cs_new:Npn \__enumext_filter_save_key_key:n #1
2284   {
2285     \str_case:nnF {#1}
2286       {
2287         { resume } {} { resume* } {} { no-store } {} { base-fix } {}
2288       }
2289       { , { \exp_not:n {#1} } }
2290   }
```

The function `\__enumext_filter_save_key_pair:nn` will be responsible for filtering the ⟨keys⟩ that are passed *"with value"* by excluding the series, resume, save-ans, save-ref, check-ans, show-ans, save-pos, wrap-ans, mark-ans, wrap-opt, save-sep, mark-ref, mini-env, mini-sep, mini-right and mini-right* keys.

```
2291 \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2292   {
2293     \str_case:nnF {#1}
2294       {
2295         { series  } {} { resume    } {} { save-ans } {} { save-ref  } {}
2296         { save-key } {} { check-ans } {} { show-ans } {} { show-pos  } {}
2297         { wrap-ans } {} { mark-ans  } {} { wrap-opt } {} { save-sep  } {}
2298         { mark-ref } {} { mini-env  } {} { mini-sep } {} { mini-right } {}
2299         { mini-right* } {}
2300       }
2301       { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2302   }
```

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

12.26.4 Function for storing content in prop list

`\__enumext_store_addto_prop:n`
`\__enumext_store_addto_prop:V`

The function `\__enumext_store_addto_prop:n` stores the content in ⟨prop list⟩ defined by save-ans key. The *"stored content"* is retrieved by means of the `\getkeyans` command.

The form in which the content is *"stored"* in the ⟨prop list⟩ is {⟨position⟩}{⟨content⟩}. This function is used by `\anskey` in enumext and enumext* environments, `\item*` in keyans and keyans* environments and `\anspic*` in keyanspic environment.

```
2303 \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2304   {
2305     \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2306       {
2307         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2308       }
2309       { #1 }
2310   }
2311 \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V, e }
```

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

12.26.5 Function for storing content in sequence

`\__enumext_store_addto_seq:n`
`\__enumext_store_addto_seq:v`
`\__enumext_store_addto_seq:V`

The function `\__enumext_store_addto_seq:n` stores the content in ⟨sequence⟩ defined by save-ans key. This function is used by `\anskey` in enumext, `\item*` in keyans and `\anspic` in keyanspic.

The form in which the content is stored in ⟨sequence⟩ is in a internal enumext or enumext* environments with the *same structure* in which the command was executed.

The *"stored content"* is retrieved by means of the `\printkeyans` command.

```
2312 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2313   {
2314     \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
2315   }
2316 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V, e }
```

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

12.26.6 Functions for storing the list structure in the sequence

`\__enumext_store_level_open:`
`\__enumext_store_level_close:`

The memorization structure of the list is handled by the functions `\__enumext_store_level_open:` and `\__enumext_store_level_close:` which are executed per level within the enumext environment.

```
2317 \cs_new_protected:Nn \__enumext_store_level_open:
2318   {
2319     \bool_if:NT \l__enumext_check_answers_bool
2320       {
2321         \tl_if_empty:cTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2322           {
2323             \__enumext_store_addto_seq:n
2324               {
2325                 \item \begin{enumext}
2326               }
2327           }
2328           {
2329             \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2330               {
```

```
2331                        \item \begin{enumext} [
2332                      }
2333                  \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2334                    {
2335                      ]
2336                    }
2337                  \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2338              }
2339          }
2340      }
2341  \cs_new_protected:Nn \__enumext_store_level_close:
2342    {
2343      \bool_if:NT \l__enumext_check_answers_bool
2344        {
2345          \__enumext_store_addto_seq:n { \end{enumext} }
2346        }
2347    }
```

(*End of definition for* __enumext_store_level_open: *and* __enumext_store_level_close:.)

__enumext_store_level_open_vii: The memorization structure of the list is handled by the functions __enumext_store_level_open_vii:
__enumext_store_level_close_vii: and __enumext_store_level_close_vii: which are executed in the enumext* environment.

```
2348  \cs_new_protected:Nn \__enumext_store_level_open_vii:
2349    {
2350      \bool_if:NT \l__enumext_check_answers_bool
2351        {
2352          \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2353            {
2354              \__enumext_store_addto_seq:n
2355                {
2356                  \item \begin{enumext*}
2357                }
2358            }
2359            {
2360              \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2361                {
2362                  \item \begin{enumext*}[
2363                }
2364              \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2365                {
2366                  ]
2367                }
2368              \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2369            }
2370        }
2371    }
2372  \cs_new_protected:Nn \__enumext_store_level_close_vii:
2373    {
2374      \bool_if:NT \l__enumext_check_answers_bool
2375        {
2376          \__enumext_store_addto_seq:n { \end{enumext*} }
2377        }
2378    }
```

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

12.26.7 Function for show marks and position

__enumext_print_keyans_box:NN   The function __enumext_print_keyans_box:NN print a box in the left margin with \l__enumext_mark_-
__enumext_print_keyans_box:cc   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

```
2379  \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2380    {
2381      \mode_leave_vertical:
2382      \skip_horizontal:n { -\dim_use:N #2 }
2383      \makebox[0pt][ r ]
2384        {
2385          \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2386            {
2387              \tl_use:N \l__enumext_mark_answer_sym_tl
```

```
2388            }
2389         }
2390      \skip_horizontal:n { \dim_use:N #2 }
2391   }
2392 \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }
```

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

12.27   The internal label and ref

The function __enumext_store_internal_ref: handles the internal *"label and ref"* system used by the save-ref and mark-ref keys for \anskey will allow to execute \ref{⟨*store name* : *position*⟩} and will return 1.(a).i.A.

__enumext_store_internal_ref: First we will remove the dots "." from the current ⟨*labels*⟩, we do not want to get double dots in our references, then we will place this in the variable \l__enumext_newlabel_arg_two_tl.

```
2393 \cs_new_protected:Nn \__enumext_store_internal_ref:
2394   {
2395      \cs_set_protected:Npn \__enumext_tmp:n ##1
2396        {
2397           \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2398           \tl_reverse:c { l__enumext_label_copy_##1_tl }
2399           \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2400           \tl_reverse:c { l__enumext_label_copy_##1_tl }
2401        }
2402      \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2403      \cs_set:Npn \__enumext_tmp:n ##1
2404        { . \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.

```
2405         \bool_lazy_all:nT
2406           {
2407              { \bool_if_p:N \g__enumext_starred_bool }
2408              { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2409           }
2410           {
2411              \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2412                { \tl_use:N \l__enumext_label_copy_vii_tl }
2413           }
2414         \bool_lazy_all:nT
2415           {
2416              { \bool_not_p:n { \g__enumext_standar_bool } }
2417              { \bool_if_p:N \l__enumext_standar_bool }
2418              { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2419           }
2420           {
2421              \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2422                {
2423                   \tl_use:N \l__enumext_label_copy_vii_tl
2424                   \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2425                }
2426           }
```

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.

```
2427         \bool_lazy_all:nT
2428           {
2429              { \bool_if_p:N \g__enumext_standar_bool }
2430              { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2431              { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2432           }
2433           {
2434              \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2435                {
2436                   \tl_use:N \l__enumext_label_copy_i_tl
2437                   \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2438                }
2439           }
2440         \cs_set:Npn \__enumext_tmp:n ##1
2441           { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } . }
2442         \bool_lazy_all:nT
```

```
2443        {
2444            { \bool_if_p:N \g__enumext_standar_bool }
2445            { \bool_if_p:N \l__enumext_starred_bool }
2446            { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2447        }
2448        {
2449            \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2450                {
2451                    \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2452                    \tl_use:N \l__enumext_label_copy_vii_tl
2453                }
2454        }
```

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

```
2455        \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2456            {
2457                \l__enumext_store_name_tl \c_colon_str
2458                \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2459            }
```

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.

```
2460        \tl_put_right:Ne \l__enumext_write_aux_file_tl
2461            {
2462                \__enumext_newlabel:nn
2463                    { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2464                    { \l__enumext_newlabel_arg_two_tl }
2465            }
2466        \l__enumext_write_aux_file_tl
2467    }
```

(*End of definition for* __enumext_store_internal_ref:.)

12.28   Common functions for \anskey **and anskey* environment**

__enumext_store_anskey_code:n   The internal function __enumext_store_anskey_code:n first we pass the ⟨*argument*⟩ to the ⟨*prop list*⟩, then checks the state of the variable \l__enumext_store_ref_key_bool handled by the save-ref key and will call the function __enumext_store_internal_ref: for the internal *"label and ref"* system. Followed by this if the show-ans or show-pos keys are active we will show the *"wrapped"* ⟨*argument*⟩.

```
2468 \cs_new_protected:Npn \__enumext_store_anskey_code:n #1
2469    {
2470        \int_gincr:N \g__enumext_item_anskey_int
2471        \__enumext_store_addto_prop:n {#1}
2472        \bool_if:NT \l__enumext_store_ref_key_bool
2473            {
2474                \__enumext_store_internal_ref:
2475            }
2476        \__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.

```
2477        \tl_clear:N \l__enumext_store_anskey_arg_tl
2478        \bool_lazy_and:nnT
2479            { \bool_if_p:N \l__enumext_store_columns_break_bool }
2480            { \bool_not_p:n { \l__enumext_starred_bool } }
2481            {
2482                \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2483            }
2484        \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.

```
2485        \bool_lazy_and:nnT
2486            { \bool_not_p:n { \l__enumext_starred_bool } }
2487            { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2488            {
2489                \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2490                    {
2491                        ( \exp_not:V \l__enumext_store_item_join_int )
2492                    }
2493                }
```

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*`.

```
2494      \bool_if:NTF \l__enumext_store_item_star_bool
2495        {
2496          \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2497          \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2498            {
2499              \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2500                {
2501                  [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2502                }
2503            }
2504          \dim_compare:nT
2505            {
2506              \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2507            }
2508            {
2509              \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2510                {
2511                  [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2512                }
2513            }
2514          \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2515        }
2516        {
2517          \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2518        }
```

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*⟩.

```
2519      \bool_lazy_and:nnT
2520        { \bool_if_p:N \l__enumext_store_ref_key_bool }
2521        { \bool_if_p:N \l__enumext_hyperref_bool }
2522        {
2523          \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2524            {
2525              \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2526                { \exp_not:V \l__enumext_mark_ref_sym_tl }
2527            }
2528        }
2529      \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2530    }
```

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

`\__enumext_anskey_show_wrap_arg:n` The function `\__enumext_anskey_show_wrap_arg:n` *"wraps"* the ⟨*argument*⟩ passed to `\anskey` and the ⟨*body*⟩ for `anskey*` when using the `wrap-ans` key.

```
2531  \cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
2532    {
2533      \par
2534      \bool_if:NTF \l__enumext_starred_bool
2535        {
2536          \__enumext_print_keyans_box:NN \l__enumext_labelwidth_vii_dim  \l__enumext_labelsep_vii_d
2537        }
2538        {
2539          \__enumext_print_keyans_box:cc
2540            { l__enumext_labelwidth_ \__enumext_level: _dim }
2541            { l__enumext_labelsep_ \__enumext_level: _dim }
2542        }
2543      \__enumext_anskey_wrapper:n { #1 }
2544    }
```

(*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.

```
2545  \cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
2546    {
2547      \bool_if:NT \l__enumext_show_answer_bool
```

```
2548        {
2549          \__enumext_anskey_show_wrap_arg:n { #1 }
2550        }
2551      \bool_if:NT \l__enumext_show_position_bool
2552        {
2553          \tl_set:Ne \l__enumext_mark_answer_sym_tl
2554            {
2555              \group_begin:
2556              \exp_not:N \normalfont
2557              \exp_not:N \footnotesize [ \int_eval:n
2558                {
2559                  \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2560                }
2561                ]
2562              \group_end:
2563            }
2564          \__enumext_anskey_show_wrap_arg:n { #1 }
2565        }
2566    }
```

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

12.29 The command \anskey

Since we will be *"storing content"* in a list environment within ⟨*sequences*⟩ and can (more or less) manage the options passed to each level, it is necessary that we have a little more control over \item when storing.

The \anskey command will cover this point and give it similar behaviour to that of \item in the enumext and enumext* environments executed as follows \anskey[⟨*key = val*⟩]{⟨*content*⟩}.

__enumext_anskey_unknown:n
__enumext_anskey_unknown:nn

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

```
2567 \keys_define:nn { enumext / anskey }
2568   {
2569     break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2570     break-col .default:n  = true,
2571     break-col .value_forbidden:n = true,
2572     item-join .int_set:N  = \l__enumext_store_item_join_int,
2573     item-join .value_required:n = true,
2574     item-star .bool_set:N = \l__enumext_store_item_star_bool,
2575     item-star .default:n  = true,
2576     item-star .value_forbidden:n = true,
2577     item-sym* .tl_set:N   = \l__enumext_store_item_symbol_tl,
2578     item-sym* .value_required:n = true,
2579     item-pos* .dim_set:N  = \l__enumext_store_item_symbol_sep_dim,
2580     item-pos* .value_required:n = true,
2581     unknown   .code:n     = { \__enumext_anskey_unknown:n {#1} },
2582   }
```

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.

```
2583 \cs_new_protected:Npn \__enumext_anskey_unknown:n #1
2584   {
2585     \exp_args:NV \__enumext_anskey_unknown:nn \l_keys_key_str {#1}
2586   }
2587 \cs_new_protected:Npn \__enumext_anskey_unknown:nn #1 #2
2588   {
2589     \tl_if_blank:nTF {#2}
2590       {
2591         \msg_error:nnn { enumext } { anskey-cmd-key-unknown } {#1}
2592       }
2593       {
2594         \msg_error:nnnn { enumext } { anskey-cmd-key-value-unknown } {#1} {#2}
2595       }
2596   }
```

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

🎯 The \anskey command will only be present when using the save-ans key in enumext and enumext* environments, otherwise it will return an error.

\anskey   We will first call the function __enumext_anskey_safe_outer: to be sure where we execute the command, then we will check the state of the variable \l__enumext_check_answers_bool set by the key no-store, if is true we will increment \g__enumext_item_anskey_int for the internal *"check answer"* system and execute the function __enumext_anskey_safe_inner:n to ensure that the command is not nested and

that the argument is not empty, finally search the $[\langle key = val \rangle]$ and call the function `\__enumext_store_-anskey_code:n`.

```
2597 \NewDocumentCommand \anskey { o +m }
2598   {
2599     \__enumext_anskey_safe_outer:
2600     \group_begin:
2601       \bool_if:NT \l__enumext_check_answers_bool
2602         {
2603           \tl_if_novalue:nF {#1}
2604             {
2605               \keys_set:nn { enumext / anskey } {#1}
2606             }
2607           \tl_if_blank:nTF {#2}
2608             {
2609               \msg_error:nn { enumext } { anskey-empty-arg }
2610             }
2611             {
2612               \__enumext_anskey_safe_inner:
2613               \__enumext_store_anskey_code:n {#2}
2614             }
2615         }
2616     \group_end:
2617   }
```

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

12.29.1 Internal functions for the command

`\__enumext_anskey_safe_outer:`
`\__enumext_anskey_safe_inner:`

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

```
2618 \cs_new_protected:Nn \__enumext_anskey_safe_outer:
2619   {
2620     \bool_if:NF \l__enumext_store_active_bool
2621       {
2622         \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
2623       }
2624     \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2625       {
2626         \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
2627       }
2628     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2629       {
2630         \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans* }
2631       }
2632     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2633       {
2634         \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
2635       }
2636   }
```

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.

```
2637 \cs_new_protected:Nn \__enumext_anskey_safe_inner:
2638   {
2639     \int_incr:N \l__enumext_anskey_level_int
2640     \int_compare:nNnT { \l__enumext_anskey_level_int } > { 1 }
2641       {
2642         \msg_error:nn { enumext } { anskey-nested }
2643       }
2644     \bool_if:NF \l__enumext_item_number_bool
2645       {
2646         \msg_error:nn { enumext } { anskey-unnumber-item }
2647       }
2648     \mode_if_math:T
2649       {
2650         \msg_error:nne { enumext } { anskey-math-mode } { \c_backslash_str anskey }
2651       }
2652   }
```

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

12.30 The environment anskey*

Managing *verbatim content* in an environment is quite complicated, I learned that when creating the `scontents` package, so to be able to have support at this point it is best to play a little with the internal code of `scontents` and *hooks*. Some considerations I should have here before implementing this:

– If some package, class or user has defined the environment with the same name somewhere in the document it would be a problem, you would not know what argument has been passed to `store-env`, if you are using the key `print-env` or the `write-out` key, sure, I can detect and modify it within the `enumext` and `enumext*` environments, but it would look strange not to have some keys available when running within these environments.

– A better (perhaps a bit paranoid) option is to define it within the environment in which the `save-ans` key is executed. and have it available only when that key is executed, here I would have absolute control of the ⟨*keys*⟩ and I make sure that `write-out` is not used, then using *hooks after* I undefine it and using *hook before* I check if it has been created by any package, class or user and I return a error, then the user will have to see how to solve the problem.

`\__enumext_undefine_anskey_env:`

The function `\__enumext_undefine_anskey_env:` will undefine the environment `anskey*` and will be passed to the function `\__enumext_execute_after_env:` (§12.31) which is executed after the environment in which the key `save-ans` is active.

```
2653 \cs_new_protected:Nn \__enumext_undefine_anskey_env:
2654   {
2655     \cs_undefine:c { anskey* }
2656     \cs_undefine:c { endanskey* }
2657     \cs_undefine:c { __scontents_anskey*_env_begin: }
2658     \cs_undefine:c { __scontents_anskey*_env_end: }
2659   }
```

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

```
2660 \__enumext_before_env:nn { enumext }
2661   {
2662     \bool_lazy_and:nnT
2663       { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2664       { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2665       {
2666         \cs_if_free:cF { __scontents_anskey*_env_begin: }
2667           {
2668             \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2669           }
2670       }
2671   }
2672 \__enumext_before_env:nn { enumext* }
2673   {
2674     \bool_lazy_and:nnT
2675       { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2676       { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2677       {
2678         \cs_if_free:cF { __scontents_anskey*_env_begin: }
2679           {
2680             \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2681           }
2682       }
2683   }
```

Detection of the `anskey*` environment inside the `keyans`, `keyans*` and `keyanspic` environments, if preceded by a not numbered `\item` or if it is in *math mode* returning the appropriate messages.

```
2684 \__enumext_before_env:nn { anskey* }
2685   {
2686     \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2687       {
2688         \msg_error:nnn { enumext } { anskey-env-wrong }{ keyans }
2689       }
2690     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2691       {
2692         \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
2693       }
2694     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2695       {
2696         \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
2697       }
2698     \bool_if:NF \l__enumext_item_number_bool
2699       {
```

```
2700          \msg_error:nn { enumext } { anskey-unnumber-item }
2701        }
2702      \mode_if_math:T
2703        {
2704          \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
2705        }
2706    }
```

(*End of definition for* __enumext_undefine_anskey_env:.)

anskey*

__enumext_anskey_env_make:n
__enumext_anskey_env_make:V
__enumext_anskey_env_define_keys:
__enumext_rescan_anskey_env:n

The function __enumext_anskey_env_make:n creates the environment anskey* (*custom version* of scontents environment) by setting the initial keys store-env={⟨*store name*⟩} and print-env=false. To maintain the *scope* of the environment and that it is only active when the key save-ans is active we will pass this function to the function __enumext_storing_exec: (§12.25.1) and we will execute it only if the variable \l__enumext_anskey_env_bool is true, with this we prevent it from being executed again when the environment is nested and the key save-ans is active, which returns an error for part of the package scontents.

```
2707 \cs_new_protected:Npn \__enumext_anskey_env_make:n #1
2708   {
2709     \bool_if:NT \l__enumext_anskey_env_bool
2710       {
2711         \newenvsc{anskey*}[store-env=#1,print-env=false]
2712         \__enumext_anskey_env_exec:
2713       }
2714   }
2715 \cs_generate_variant:Nn \__enumext_anskey_env_make:n { V }
```

The function __enumext_anskey_env_define_keys: will add the keys break-col, item-join, item-join, item-star, item-sym* and item-pos* and will leave the keys print-env, store-env and write-out undefined. We will apply this function using the *hook* function __enumext_before_env:nn.

```
2716 \cs_new_protected:Nn \__enumext_anskey_env_define_keys:
2717   {
2718     \keys_define:nn { scontents / scontents }
2719       {
2720         break-col .bool_gset:N = \g__enumext_store_columns_break_bool,
2721         break-col .default:n   = true,
2722         break-col .value_forbidden:n = true,
2723         item-join .int_gset:N  = \g__enumext_store_item_join_int,
2724         item-join .value_required:n = true,
2725         item-star .bool_gset:N = \g__enumext_store_item_star_bool,
2726         item-star .default:n   = true,
2727         item-star .value_forbidden:n = true,
2728         item-sym* .tl_gset:N   = \g__enumext_store_item_symbol_tl,
2729         item-sym* .value_required:n = true,
2730         item-pos* .dim_gset:N  = \g__enumext_store_item_symbol_sep_dim,
2731         item-pos* .value_required:n = true,
2732         print-env .undefine:,
2733         store-env .undefine:,
2734         write-out .undefine:,
2735         unknown   .code:n      = { \__enumext_anskey_env_unknown:n {##1} },
2736       }
2737   }
```

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.

```
2738 \cs_new_protected:Npn \__enumext_anskey_env_unknown:n #1
2739   {
2740     \exp_args:NV \__enumext_anskey_env_unknown:nn \l_keys_key_str {#1}
2741   }
2742 \cs_new_protected:Npn \__enumext_anskey_env_unknown:nn #1#2
2743   {
2744     \tl_if_blank:nTF {#2}
2745       {
2746         \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
2747       }
2748       {
2749         \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
2750       }
2751   }
```

The function `\__enumext_anskey_env_reset_keys:` will leave the keys break-col, item-join, item-join, item-star, item-sym* and item-pos* undefined. We will apply this function using the *hook* function `\__enumext_after_env:nn`.

```
2752 \cs_new_protected:Nn \__enumext_anskey_env_reset_keys:
2753   {
2754     \keys_define:nn { scontents / scontents }
2755       {
2756         break-col .undefine:,
2757         item-join .undefine:,
2758         item-star .undefine:,
2759         item-sym* .undefine:,
2760         item-pos* .undefine:,
2761         write-out .code:n    = {
2762                                 \bool_set_false:N \l__scontents_storing_bool
2763                                 \bool_set_true:N  \l__scontents_writing_bool
2764                                 \tl_set:Nn \l__scontents_fname_out_tl {##1}
2765                               },
2766         write-out .value_required:n = true,
2767         print-env .meta:nn   = { scontents } { print-env = ##1 },
2768         print-env .default:n = true,
2769         store-env .meta:nn   = { scontents } { store-env = ##1 },
2770         unknown   .code:n    = { \__scontents_parse_environment_keys:n {##1} },
2771       }
2772   }
```

The function `\__enumext_rescan_anskey_env:n` will be responsible for bringing the ⟨*body*⟩ of the environment saved in the sequence `\g__scontents_name_`⟨*store name*⟩`_seq` to pass it to our *sequence and prop list*.

```
2773 \cs_new_protected:Npn \__enumext_rescan_anskey_env:n #1
2774   {
2775     \group_begin:
2776       \int_set:Nn \tex_newlinechar:D { `\^^J }
2777       \__scontents_rescan_tokens:x
2778         {
2779           \endgroup % This assumes \catcode`\\=0... Things might go off otherwise.
2780           #1
2781         }
2782   }
```

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

`\__enumext_anskey_env_exec:` The function `\__enumext_anskey_env_exec:` will be responsible for processing all the code necessary for the execution of the environment. The first thing will be to add our ⟨*keys*⟩.

```
2783 \cs_new_protected:Nn \__enumext_anskey_env_exec:
2784   {
2785     \__enumext_before_env:nn { anskey* }
2786       {
2787         \__enumext_anskey_env_define_keys:
2788       }
```

Now we will execute our actions after the anskey* environment is closed. We'll fetch the contents of the *environment body* that is now saved in `\g__scontents_name_`⟨*store name*⟩`_seq` and store it in the variable `\l__enumext_store_anskey_env_tl` then we execute the rest of the functions.

```
2789     \hook_if_empty:nF {env/anskey*/after}
2790       {
2791         \hook_gremove_code:nn {env/anskey*/after} { * }
2792       }
2793     \__enumext_after_env:nn { anskey* }
2794       {
2795         \__enumext_anskey_env_save_keys:
2796         \tl_clear:N \l__enumext_store_anskey_env_tl
2797         \tl_clear:N \l__enumext_store_anskey_opt_tl
2798         \bool_if:NT \l__enumext_check_answers_bool
2799           {
2800             \tl_gset:Ne \l__enumext_store_anskey_env_tl
2801               {
2802                 \seq_item:ce { g__scontents_name_ \l__enumext_store_name_tl _seq } { -1 }
2803               }
2804             \regex_match:nVTF
2805               { ^\s* \z | ^\s* \u{c__scontents_hidden_space_str} \z }
2806               \l__enumext_store_anskey_env_tl
```

```
2807                       {
2808                           \msg_error:nn { enumext } { anskey-empty-arg }
2809                       }
2810                       {
2811                           \__enumext_anskey_env_store:
2812                       }
2813                   }
2814               \__enumext_anskey_env_clean_vars:
2815               \__enumext_anskey_env_reset_keys:
2816           }
2817       }
```

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

(*End of definition for* __enumext_anskey_env_exec:.)

__enumext_anskey_env_save_keys:
__enumext_anskey_env_store:
__enumext_anskey_env_clean_vars:

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

```
2818  \cs_new_protected:Nn \__enumext_anskey_env_save_keys:
2819    {
2820       \bool_lazy_and:nnT
2821         { \bool_if_p:N \g__enumext_store_columns_break_bool }
2822         { \bool_not_p:n { \l__enumext_starred_bool } }
2823         {
2824           \tl_put_left:Ne \l__enumext_store_anskey_opt_tl { ,break-col, }
2825         }
```

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

```
2826       \bool_lazy_and:nnT
2827         { \bool_not_p:n { \l__enumext_starred_bool } }
2828         { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }
2829         {
2830           \tl_put_left::Ne \l__enumext_store_anskey_opt_tl
2831             {
2832               ,item-join = \exp_not:V \g__enumext_store_item_join_int,
2833             }
2834         }
```

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

```
2835       \bool_if:NT \g__enumext_store_item_star_bool
2836         {
2837           \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2838             {
2839               ,item-star,
2840             }
2841           \tl_if_empty:NF \g__enumext_store_item_symbol_tl
2842             {
2843               \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2844                 {
2845                   ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,
2846                 }
2847             }
2848           \dim_compare:nT
2849             {
2850               \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
2851             }
2852             {
2853               \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2854                 {
2855                   ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
2856                 }
2857             }
2858         }
2859    }
```

The function __enumext_anskey_env_store: will be responsible for storing the content of the environ-
ment using the functions __enumext_store_anskey_code:n and __enumext_rescan_anskey_env:n.

```
2860  \cs_new_protected:Nn \__enumext_anskey_env_store:
```

```
2861    {
2862        \group_begin:
2863            \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
2864                {
2865                    \exp_args:Ne
2866                        \__enumext_store_anskey_code:n
2867                            {
2868                                \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2869                            }
2870                }
2871                {
2872                    \keys_set_known:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
2873                    \exp_args:Ne
2874                        \__enumext_store_anskey_code:n
2875                            {
2876                                \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
2877                            }
2878                }
2879        \group_end:
2880    }
```

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

```
2881    \cs_new_protected:Nn \__enumext_anskey_env_clean_vars:
2882        {
2883            \bool_gset_false:N \g__enumext_store_columns_break_bool
2884            \int_gzero:N        \g__enumext_store_item_join_int
2885            \bool_gset_false:N \g__enumext_store_item_star_bool
2886            \tl_gclear:N        \g__enumext_store_item_symbol_tl
2887            \dim_gzero:N        \g__enumext_store_item_symbol_sep_dim
2888        }
```

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

12.31   Executing anskey*, check-ans and write .log

`\__enumext_execute_after_env:`   The `\__enumext_execute_after_env:` function will first return the appropriate message for the end of the environment in which the save-ans key is being executed, then call the `\__enumext_item_answer_diff:` function and then will write the values of the global variables used to the .log file. If the key check-ans is active it will execute the function `\__enumext_check_ans_show:` and show the result in the terminal, otherwise it will execute the function `\__enumext_check_ans_log:` and write the results in the .log file, undefine the environment anskey* (§12.30) through the function `\__enumext_undefine_anskey_env:` and finally we execute the function `\__enumext_reset_global_vars:` returning the used variables to their original state.

```
2889    \cs_new_protected:Nn \__enumext_execute_after_env:
2890        {
2891            \int_compare:nNnT { \l__enumext_level_int } = { 0 }
2892                {
2893                    \tl_if_empty:NF \g__enumext_store_name_tl
2894                        {
2895                            \__enumext_stop_save_ans_msg:
2896                            \__enumext_item_answer_diff:
2897                            \__enumext_log_global_vars:
2898                            \__enumext_log_answer_vars:
2899                            \bool_if:NTF \g__enumext_check_ans_key_bool
2900                                {
2901                                    \__enumext_check_ans_show:
2902                                }
2903                                { \__enumext_check_ans_log: }
2904                            \__enumext_undefine_anskey_env:
2905                        }
2906                    \__enumext_reset_global_vars:
2907                }
2908        }
```

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

🎯 This function is passed to the function `\__enumext_after_env:nn` for the environments enumext (§12.38) and enumext* (§12.43) and it is executed only when the environments are not nested or at some level of these..

12.32   Common functions for `keyans`, `keyans*` and `keyanspic`

12.32.1   Storing content in prop list

__enumext_keyans_addto_prop:n

The function `\__enumext_keyans_addto_prop:n` will pass the contents of the current ⟨*label*⟩ `\l__-enumext_label_v_tl` for the `keyans` environment and the current ⟨*label*⟩ `\l__enumext_label_vi_tl` for the `keyanspic` environment when using `\item*` and `\anspic*`, followed by the *contents* of the *optional argument* of both commands to the `\l__enumext_store_current_label_tl` variable, which will be passed to the ⟨*prop list*⟩ defined by the `save-ans` key using the `\__enumext_store_addto_prop:V`.

```
2909  \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
2910    {
2911      \tl_clear:N \l__enumext_store_current_label_tl
2912      \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2913        {
2914          \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
2915        }
2916        {
2917          \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
2918        }
2919      \tl_if_novalue:nF { #1 }
2920        {
2921          % Set save-sep
2922          \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2923            {
2924              \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_store_keyans_item_op
2925            }
2926          \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
2927        }
2928      \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
2929    }
```

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

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

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

__enumext_keyans_store_ref:
__enumext_keyans_store_ref_aux_i:
__enumext_keyans_store_ref_aux_ii:

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

```
2930  \cs_new_protected:Nn \__enumext_keyans_store_ref:
2931    {
2932      \bool_if:NT \l__enumext_store_ref_key_bool
2933        {
2934          \cs_set_protected:Npn \__enumext_tmp:n ##1
2935            {
2936              \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2937              \tl_reverse:c { l__enumext_label_copy_##1_tl }
2938              \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2939              \tl_reverse:c { l__enumext_label_copy_##1_tl }
2940            }
2941          \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
2942          \__enumext_keyans_store_ref_aux_i:
2943        }
2944    }
```

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`.

```
2945  \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
2946    {
2947      \bool_if:NT \g__enumext_starred_bool
2948        {
2949          \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
2950        }
2951      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2952        {
2953          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2954            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
```

```
2955        }
2956      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2957        {
2958          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2959            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
2960        }
2961      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2962        {
2963          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2964            { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
2965        }
2966      \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2967        {
2968          \l__enumext_store_name_tl \c_colon_str
2969          \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2970        }
2971      \__enumext_keyans_store_ref_aux_ii:
2972    }
```

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.

```
2973  \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
2974    {
2975      \tl_put_right:Ne \l__enumext_write_aux_file_tl
2976        {
2977          \__enumext_newlabel:nn
2978            { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2979            { \l__enumext_newlabel_arg_two_tl }
2980        }
2981      \l__enumext_write_aux_file_tl
2982    }
```

(*End of definition for* __enumext_keyans_store_ref:, __enumext_keyans_store_ref_aux_i:, *and* __enumext_keyans_-store_ref_aux_ii:.)

12.32.3 Storing content in sequence

__enumext_keyans_addto_seq:n
__enumext_keyans_addto_seq_link:

The function __enumext_keyans_addto_seq:n will pass the contents of the current ⟨*label*⟩ \l__-enumext_label_v_tl for the keyans environment and the \l__enumext_label_vi_tl for the keyanspic environment when using \item* and \anspic*, followed by the ⟨*contents*⟩ of the *optional argument* of both commands to the \l__enumext_store_current_label_tl variable to the sequence defined by the save-ans key.

```
2983  \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
2984    {
2985      \tl_clear:N \l__enumext_store_current_label_tl
2986      \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
2987        {
2988          \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
2989        }
2990        {
2991          \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
2992        }
2993      \tl_if_novalue:nF { #1 }
2994        {
2995          \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
2996            {
2997              \tl_put_right:Ne \l__enumext_store_current_label_tl
2998                {
2999                  \l__enumext_store_keyans_item_opt_sep_tl
3000                }
3001            }
3002          \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
3003        }
3004      \__enumext_keyans_addto_seq_link:
3005    }
```

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.

```
3006  \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
3007    {
3008      \bool_lazy_and:nnT
3009        { \bool_if_p:N \l__enumext_store_ref_key_bool }
3010        { \bool_if_p:N \l__enumext_hyperref_bool }
3011        {
3012          \tl_put_right:Ne \l__enumext_store_current_label_tl
3013            {
3014              \hfill \exp_not:N \hyperlink
3015                {
3016                  \exp_not:V \l__enumext_newlabel_arg_one_tl
3017                }
3018                { \exp_not:V \l__enumext_mark_ref_sym_tl }
3019            }
3020        }
3021      \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
3022      \bool_if:NT \l__enumext_check_answers_bool
3023        {
3024          \int_gincr:N \g__enumext_item_anskey_int
3025        }
3026    }
```

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

12.32.4   The show-ans and show-pos keys for keyans and keyanspic

The code is very similar to the \anskey code, but, if I change the order of the operations the counter off ⟨*label*⟩ are incorrect.

__enumext_keyans_show_left:n
__enumext_keyans_show_ans:
__enumext_keyans_show_pos:
__enumext_keyans_show_item_opt:

Common function to show *starred commands* \item* and ⟨*position*⟩ of stored content in ⟨*prop list*⟩ for keyans and keyanspic. Need add 1 to \g__enumext_⟨*store name*⟩_prop for show-pos key.

```
3027  \cs_new_protected:Npn \__enumext_keyans_show_left:n #1
3028    {
3029      \tl_if_novalue:nF { #1 }
3030        {
3031          \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
3032        }
3033      \bool_if:NT \l__enumext_show_answer_bool
3034        {
3035          \__enumext_keyans_show_ans:
3036        }
3037      \bool_if:NT \l__enumext_show_position_bool
3038        {
3039          \__enumext_keyans_show_pos:
3040        }
3041    }
3042  \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
3043    {
3044      \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
3045        {
3046          \bool_lazy_or:nnT
3047            { \bool_if_p:N \l__enumext_show_answer_bool }
3048            { \bool_if_p:N \l__enumext_show_position_bool }
3049            {
3050              \__enumext_keyans_wrapper_opt:n { \l__enumext_store_current_opt_arg_tl } \c_space_tl
3051            }
3052        }
3053    }
3054  \cs_new_protected:Nn \__enumext_keyans_show_ans:
3055    {
3056      \bool_if:NT \l__enumext_starred_bool
3057        {
3058          \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
3059          \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
3060        }
3061      \tl_put_left:Nn \l__enumext_label_v_tl
3062        {
3063          \__enumext_print_keyans_box:NN
3064            \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3065        }
3066    }
3067  \cs_new_protected:Nn \__enumext_keyans_show_pos:
```

```
3068    {
3069      \bool_if:NT \l__enumext_starred_bool
3070        {
3071          \dim_set_eq:NN \l__enumext_labelwidth_i_dim \l__enumext_labelwidth_vii_dim
3072          \dim_set_eq:NN \l__enumext_labelsep_i_dim \l__enumext_labelsep_vii_dim
3073        }
3074      \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
3075        {
3076          \tl_set:Ne \l__enumext_mark_answer_sym_tl
3077            {
3078              \group_begin:
3079              \exp_not:N \normalfont
3080              \exp_not:N \footnotesize [ \int_eval:n
3081                {
3082                  \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
3083                }
3084                ]
3085              \group_end:
3086            }
3087        }
3088        {
3089          \tl_set:Ne \l__enumext_mark_answer_sym_tl
3090            {
3091              \group_begin:
3092              \exp_not:N \normalfont
3093              \exp_not:N \footnotesize [ \int_eval:n
3094                {
3095                  \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1
3096                }
3097                ]
3098              \group_end:
3099            }
3100        }
3101      \tl_put_left:Nn \l__enumext_label_v_tl
3102        {
3103          \__enumext_print_keyans_box:NN
3104            \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
3105        }
3106    }
```

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

12.33 Redefining `\item` and `\makelabel` in enumext

Redefining the `\item` command is not as simple as I thought. This command works in conjunction with the `\makelabel` command so I have to redefine both of them, in addition to this, we will have to use a couple of *global* variables to pass the values from one command to the other.

The `\item` and `\item[`⟨*custom*⟩`]` commands work in the usual way on enumext and we will add `\item*`, `\item*[`⟨*symbol*⟩`]` and `\item*[`⟨*symbol*⟩`][`⟨*offset*⟩`]`.

`\__enumext_default_item:n`   First we will see if the *optional argument* is present, if it is NOT present we will check the state of the variable `\l__enumext_check_answers_bool` set by the key `no-store`, set the boolean variable `\l__enumext_-wrap_label_X_bool` to "true" for the key `wrap-label` and execute `\__enumext_item_std:w` and the key `itemindent`, otherwise we will check the state of the boolean variable `\l__enumext_wrap_label_opt_-X_bool` set by the key `wrap-label*` and execute `\__enumext_item_std:w` with the *optional argument* and the key `itemindent`.

```
3107  \cs_new_protected:Npn \__enumext_default_item:n #1
3108    {
3109      \tl_if_novalue:nTF {#1}
3110        {
3111          \bool_if:NT \l__enumext_check_answers_bool
3112            {
3113              \int_gincr:N \g__enumext_item_number_int
3114              \bool_set_true:N \l__enumext_item_number_bool
3115            }
3116          \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
3117          \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3118        }
3119        {
3120          \bool_set_eq:cc
3121            { l__enumext_wrap_label_ \__enumext_level: _bool }
```

```
3122                { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
3123            \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl
3124        }
3125    }
```

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

__enumext_starred_item:nn
__enumext_item_star_exec:

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

#1: \l__enumext_item_symbol_X_tl
#2: \l__enumext_item_symbol_sep_X_dim

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

```
3126  \cs_new_protected:Npn \__enumext_starred_item:nn #1 #2
3127    {
3128      \tl_if_novalue:nTF {#1}
3129        {
3130          \tl_gset_eq:Nc
3131            \g__enumext_item_symbol_aux_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
3132        }
3133        {
3134          \tl_gset:Nn \g__enumext_item_symbol_aux_tl {#1}
3135        }
3136      \tl_if_novalue:nTF {#2}
3137        {
3138          \dim_set_eq:cc
3139            { l__enumext_item_symbol_sep_ \__enumext_level: _dim }
3140            { l__enumext_labelsep_ \__enumext_level: _dim }
3141        }
3142        {
3143          \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
3144        }
3145      \bool_if:NT \l__enumext_check_answers_bool
3146        {
3147          \int_gincr:N \g__enumext_item_number_int
3148          \bool_set_true:N \l__enumext_item_number_bool
3149        }
3150      \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
3151      \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3152    }
```

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

```
3153  \cs_new_protected:Nn \__enumext_item_star_exec:
3154    {
3155      \tl_if_empty:cF { l__enumext_item_symbol_ \__enumext_level: _tl }
3156        {
3157          \mode_leave_vertical:
3158          \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3159          \hbox_overlap_left:n { \g__enumext_item_symbol_aux_tl }
3160          \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3161        }
3162    }
```

(*End of definition for* __enumext_starred_item:nn *and* __enumext_item_star_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 (§12.38).

```
3163  \cs_new_protected:Nn \__enumext_redefine_item:
3164    {
3165      \RenewDocumentCommand \item { s o o }
3166        {
3167          \bool_if:nTF {##1}
3168            {
```

```
3169                     \__enumext_starred_item:nn {##2} {##3}
3170                 }
3171                 { \__enumext_default_item:n {##2} }
3172         }
3173     }
```

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

🔷 When *tagged* PDF is active \makelabel is redefined as \hss #1 and the only way to get the align key to work correctly is by using \makebox. The solution here is to redefine \makelabel conditionally using \IfDocumentMetadataTF.

__enumext_make_label:
__enumext_make_label_std:
__enumext_make_label_box:

The function __enumext_make_label: redefine \makelabel for the keys 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 (§12.38).

```
3174 \cs_new_protected:Nn \__enumext_make_label:
3175   {
3176     \IfDocumentMetadataTF
3177       {
3178         \__enumext_make_label_box:
3179       }
3180       { \__enumext_make_label_std: }
3181   }
```

Standard definition when \DocumentMetadata is not active.

```
3182 \cs_new_protected:Nn \__enumext_make_label_std:
3183   {
3184     \RenewDocumentCommand \makelabel { m }
3185       {
3186         \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
3187         \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3188         \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3189           {
3190             \__enumext_item_star_exec:
3191             \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3192           }
3193           { ##1 }
3194         \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
3195         \tl_gclear:N \g__enumext_item_symbol_aux_tl
3196       }
3197   }
```

Definition using \makebox when \DocumentMetadata is active.

```
3198 \cs_new_protected:Nn \__enumext_make_label_box:
3199   {
3200     \RenewDocumentCommand \makelabel { m }
3201       {
3202         \makebox
3203         [ \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim } ]
3204         [ \str_use:c { l__enumext_align_label_pos_ \__enumext_level: _str } ]
3205           {
3206             \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3207             \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3208               {
3209                 \__enumext_item_star_exec:
3210                 \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3211               }
3212               { ##1 }
3213             \tl_gclear:N \g__enumext_item_symbol_aux_tl
3214           }
3215       }
3216   }
```

(*End of definition for* __enumext_make_label:*,* __enumext_make_label_std:*, and* __enumext_make_label_box:*.*)

12.34   Setting item-sym* and item-pos* keys

In order to have a cleaner implementation of \item* for the enumext and enumext* environments it is best to define a couple of keys that allow us to control and set by default the ⟨*symbol*⟩ and its ⟨*offset*⟩.

item-sym*
item-pos*

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

```
3217 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
3218   {
3219     \keys_define:nn { enumext / #1 }
```

```
3220          {
3221             item-sym* .tl_set:c  = { l__enumext_item_symbol_#2_tl },
3222             item-sym* .value_required:n = true,
3223             item-sym* .initial:n  = {$\star$},
3224             item-pos* .dim_set:c  = { l__enumext_item_symbol_sep_#2_dim },
3225             item-pos* .value_required:n = true,
3226          }
3227       }
3228    \clist_map_inline:nn
3229       {
3230          {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
3231       }
3232       { \__enumext_tmp:nn #1 }
```

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

12.35   Handling unknown keys

At this point in the code I already know that I will not add more ⟨*keys*⟩ and since I have already been quite *paranoid and restrictive* with the definitions of environments and commands, the only thing left to do is do it with the ⟨*keys*⟩ (you have to be consistent in life).

12.35.1   Handling unknown keys for keyans and keyans*

unknown
__enumext_keyans_unknown_keys:n
__enumext_keyans_unknown_keys:nn

Define and set unknown key for keyans and keyans* environments.

```
3233    \cs_set_protected:Npn \__enumext_tmp:n #1
3234       {
3235          \keys_define:nn { enumext / #1 }
3236             {
3237                unknown .code:n = { \__enumext_keyans_unknown_keys:n {##1} }
3238             }
3239       }
3240    \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }
```

Internal functions for handling unknown key.

```
3241    \cs_new_protected:Npn \__enumext_keyans_unknown_keys:n #1
3242       {
3243          \exp_args:NV \__enumext_keyans_unknown_keys:nn \l_keys_key_str {#1}
3244       }
3245    \cs_new_protected:Npn \__enumext_keyans_unknown_keys:nn #1#2
3246       {
3247          \tl_if_blank:nTF {#2}
3248             {
3249                \msg_error:nnn { enumext } { keyans-unknown-key } {#1}
3250             }
3251             {
3252                \msg_error:nnnn { enumext } { keyans-unknown-key-value } {#1} {#2}
3253             }
3254       }
```

(*End of definition for* unknown *,* __enumext_keyans_unknown_keys:n *, and* __enumext_keyans_unknown_keys:nn*.*)

12.35.2   Handling unknown keys for enumext*

unknown
__enumext_starred_unknown_keys:n
__enumext_starred_unknown_keys:nn

Define and set unknown key for enumext* environment.

```
3255    \keys_define:nn { enumext / enumext* }
3256       {
3257          unknown .code:n = { \__enumext_starred_unknown_keys:n {#1} }
3258       }
```

Internal functions for handling unknown key.

```
3259    \cs_new_protected:Npn \__enumext_starred_unknown_keys:n #1
3260       {
3261          \exp_args:NV \__enumext_starred_unknown_keys:nn \l_keys_key_str {#1}
3262       }
3263    \cs_new_protected:Npn \__enumext_starred_unknown_keys:nn #1#2
3264       {
3265          \tl_if_blank:nTF {#2}
3266             {
3267                \msg_error:nnn { enumext } { starred-unknown-key } {#1}
3268             }
3269             {
3270                \msg_error:nnnn { enumext } { starred-unknown-key-value } {#1} {#2}
3271             }
3272       }
```

(*End of definition for* unknown *,* __enumext_starred_unknown_keys:n *, and* __enumext_starred_unknown_keys:nn*.*)

12.35.3 Handling unknown keys for enumext

Defines and set the key unknown for enumext environment.

```
3273 \cs_set_protected:Npn \__enumext_tmp:n #1
3274   {
3275     \keys_define:nn { enumext / #1 }
3276       {
3277         unknown .code:n = { \__enumext_standar_unknown_keys:n {##1} }
3278       }
3279   }
3280 \clist_map_inline:nn { level-1,level-2,level-3,level-4 } { \__enumext_tmp:n {#1} }
```

Internal functions for handling unknown key.

```
3281 \cs_new_protected:Npn \__enumext_standar_unknown_keys:n #1
3282   {
3283     \exp_args:NV \__enumext_standar_unknown_keys:nn \l_keys_key_str {#1}
3284   }
3285 \cs_new_protected:Npn \__enumext_standar_unknown_keys:nn #1#2
3286   {
3287     \tl_if_blank:nTF {#2}
3288       {
3289         \msg_error:nnn { enumext } { standar-unknown-key } {#1}
3290       }
3291       {
3292         \msg_error:nnnn { enumext } { standar-unknown-key-value } {#1} {#2}
3293       }
3294   }
```

(*End of definition for* unknown *,* __enumext_standar_unknown_keys:n *, and* __enumext_standar_unknown_keys:nn*.*)

12.36 Redefining \item and \makelabel in keyans

The \item and \item[⟨*custom*⟩] commands work in the usual way in keyans, but the \item* and \item*[⟨*content*⟩] commands *store* the current ⟨*label*⟩ next to the ⟨*content*⟩ if it is present in the ⟨*sequence*⟩ and ⟨*prop list*⟩ defined by save-ans key.

The function __enumext_keyans_default_item:n executes the original behavior of the \item along with the keys wrap-label, wrap-label* and itemindent.

```
3295 \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
3296   {
3297     \tl_if_novalue:nTF { #1 }
3298       {
3299         \bool_set_true:N \l__enumext_wrap_label_v_bool
3300         \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
3301       }
3302       {
3303         \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
3304         \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
3305       }
3306   }
```

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

The function __enumext_keyans_starred_item:n which will make a temporary copy of the current ⟨*label*⟩, execute the show-ans or show-pos keys using the function __enumext_keyans_show_left:n and will display the ⟨*contents*⟩ of that item using the internal copy __enumext_item_std:w, this is necessary to prevent incrementing the current *"counter"* of the original ⟨*label*⟩, followed by this it will execute function __enumext_keyans_show_item_opt: handled by wrap-opt key.

```
3307 \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
3308   {
3309     \tl_set_eq:NN \l__enumext_store_current_label_tmp_tl \l__enumext_label_v_tl
3310     \__enumext_keyans_show_left:n { #1 }
3311     \bool_set_true:N \l__enumext_wrap_label_v_bool
3312     \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
3313     \__enumext_keyans_show_item_opt:
```

Recover the original value of the current ⟨*label*⟩ and *store* it first in the ⟨*prop list*⟩ (including the *optional argument*), run the internal *"label and ref"* system if the save-ref key is active, *store* it in the ⟨*sequence*⟩ and finally increments \g__enumext_check_starred_cmd_int for internal check system.

```
3314     \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_store_current_label_tmp_tl
3315     \__enumext_keyans_addto_prop:n { #1 }
```

```
3316       \__enumext_keyans_store_ref:
3317       \__enumext_keyans_addto_seq:n { #1 }
3318       \int_gincr:N \g__enumext_check_starred_cmd_int
3319     }
```

(*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 need to use \peek_remove_spaces:n to prevent an unwanted space when using \item* in conjunction with the itemindent key. This function are passed to __enumext_list_arg_two_v: used in the definition of the keyans environment (§12.37.2).

```
3320  \cs_new_protected:Nn \__enumext_keyans_redefine_item:
3321    {
3322      \RenewDocumentCommand \item { s o }
3323        {
3324          \bool_if:nTF {##1}
3325            {
3326              \peek_remove_spaces:n
3327                {
3328                  \__enumext_keyans_starred_item:n {##2}
3329                }
3330            }
3331            {
3332              \__enumext_keyans_default_item:n {##2}
3333            }
3334        }
3335    }
```

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

__enumext_keyans_make_label:
__enumext_keyans_make_label_std:
__enumext_keyans_make_label_box:

The function __enumext_keyans_make_label: redefine \makelabel for the keys align, font, wrap-label, wrap-label* and \item* for keyans environment. This function are passed to __enumext_list_arg_two_v: used in the definition of the keyans environment (§12.37.2).

```
3336  \cs_new_protected:Nn \__enumext_keyans_make_label:
3337    {
3338      \IfDocumentMetadataTF
3339        {
3340          \__enumext_keyans_make_label_box:
3341        }
3342        { \__enumext_keyans_make_label_std: }
3343    }
```

Standard definition when \DocumentMetadata is not active.

```
3344  \cs_new_protected:Nn \__enumext_keyans_make_label_std:
3345    {
3346      \RenewDocumentCommand \makelabel { m }
3347        {
3348          \tl_use:N \l__enumext_label_fill_left_v_tl
3349          \tl_use:N \l__enumext_label_font_style_v_tl
3350          \bool_if:NTF \l__enumext_wrap_label_v_bool
3351            {
3352              \__enumext_wrapper_label_v:n { ##1 }
3353            }
3354            { ##1 }
3355          \tl_use:N \l__enumext_label_fill_right_v_tl
3356        }
3357    }
```

Definition using \makebox when \DocumentMetadata is active.

```
3358  \cs_new_protected:Nn \__enumext_keyans_make_label_box:
3359    {
3360      \RenewDocumentCommand \makelabel { m }
3361        {
3362          \makebox[ \l__enumext_labelwidth_v_dim ][ \l__enumext_align_label_pos_v_str ]
3363            {
3364              \tl_use:N \l__enumext_label_font_style_v_tl
3365              \bool_if:NTF \l__enumext_wrap_label_v_bool
3366                {
3367                  \__enumext_wrapper_label_v:n { ##1 }
3368                }
```

```
3369                        { ##1 }
3370                    }
3371                }
3372        }
```

(*End of definition for* __enumext_keyans_make_label:, __enumext_keyans_make_label_std:, *and* __enumext_keyans_make_label_box:.)

12.37 Second argument of the lists

At this point of the code we have already programmed most the necessary tools to create a custom `list` environment, remember that the function __enumext_start_list:nn takes two arguments, the first one we have ready, the second one we will define for all the levels of the environment `enumext` and the environment `keyans`.

12.37.1 Calculation of \leftmargin and \itemindent

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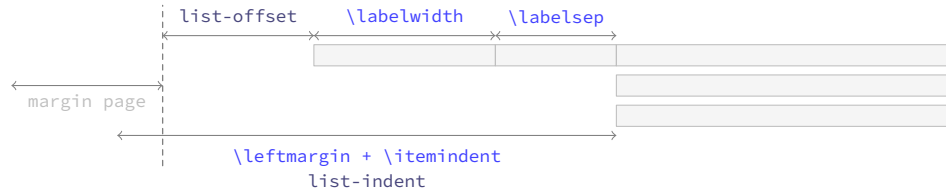
Figure 9: Representation of standard horizontal lengths in `list` environment.

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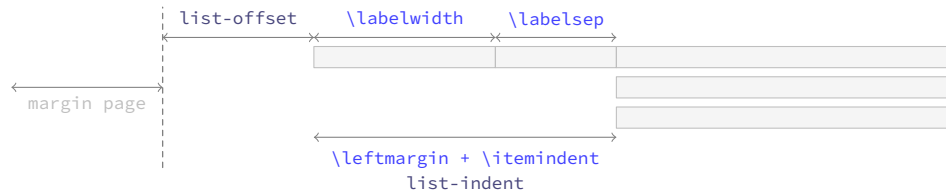
Figure 10: Representation of horizontal lengths concept in list in enumext.

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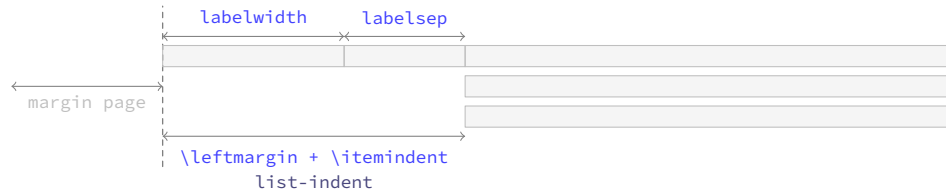
Figure 11: Default horizontal lengths in enumext.

__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 (§12.37.2).

```
3373  \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
3374    {
3375      \dim_compare:nNnT { #1 } < { \c_zero_dim }
3376        {
3377          \msg_warning:nnnV { enumext } { width-non-positive }{ labelwidth }{ #1 }
3378          \dim_set:Nn #1 { \dim_abs:n { #1 } }
3379        }
3380      \dim_compare:nNnT { #2 } < { \c_zero_dim }
3381        {
3382          \msg_warning:nnnV { enumext } { width-negative }{ labelsep }{ #2 }
3383          \dim_set:Nn #2 { \dim_abs:n { #2 } }
3384        }
```

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

```
3385        \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2 } }
```

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

```
3386        \dim_compare:nNnTF { #4 } < { \c_zero_dim }
3387          {
3388            \dim_set:Nn #6 { #1 + #2 - #4}
3389            \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3390          }
3391          {
3392            \dim_compare:nNnT { #4 } = { #1 + #2 }
3393              { \dim_set:Nn #6 { \c_zero_dim } }
3394            \dim_compare:nNnT { #4 } < { #1 + #2 }
3395              { \dim_set:Nn #6 { #1 + #2 - #4} }
3396            \dim_compare:nNnT { #4 } > { #1 + #2 }
3397              {
3398                \dim_set:Nn #6 { -#1 - #2 + #4}
3399                \dim_set:Nn #6 { #6*-1}
3400              }
3401            \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3402          }
3403      }
3404  \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { ccccccc }
```

(*End of definition for* `\__enumext_calc_hspace:NNNNNNN`.)

12.37.2 Setting second argument of the lists

`\__enumext_list_arg_two_i:`
`\__enumext_list_arg_two_ii:`
`\__enumext_list_arg_two_iii:`
`\__enumext_list_arg_two_iv:`
`\__enumext_list_arg_two_v:`

We will "not set" `\leftmargini`, `\leftmarginii`, `\leftmarginiii` or `\leftmarginiv`, in this case, we will directly set the parameters for vertical and horizontal list spacing per level.

```
3405  \cs_set_protected:Npn \__enumext_tmp:n #1
3406    {
3407      \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3408        {
3409          \__enumext_calc_hspace:ccccccc
3410            { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3411            { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3412            { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3413            { l__enumext_leftmargin_tmp_#1_bool }
3414          \clist_map_inline:nn
3415            { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3416            { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
3417          \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3418            { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
3419          \usecounter { enumX#1 }
3420          \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
3421          \str_if_eq:nnTF {#1} { v }
3422            {
3423              \__enumext_keyans_redefine_item:
3424              \__enumext_keyans_make_label:
3425              \__enumext_keyans_ref:
3426              \__enumext_keyans_fake_item_indent:
3427              \bool_if:cT { l__enumext_show_length_#1_bool }
3428                {
3429                  \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3430                }
3431            }
3432            {
3433              \__enumext_redefine_item:
3434              \__enumext_make_label:
3435              \__enumext_standar_ref:
3436              \__enumext_fake_item_indent:
3437              \bool_if:cT { l__enumext_show_length_#1_bool }
3438                {
3439                  \msg_term:nnne { enumext } { list-lengths } {#1}
3440                    { \int_use:N \l__enumext_level_int }
3441                }
3442            }
3443        }
3444    }
3445  \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.

```
3446 \cs_set_protected:Npn \__enumext_tmp:n #1
3447   {
3448     \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3449       {
3450         \bool_set_true:c { l__enumext_leftmargin_tmp_#1_bool }
3451         \dim_zero:c { l__enumext_leftmargin_tmp_#1_dim }
3452         \__enumext_calc_hspace:ccccccc
3453           { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3454           { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3455           { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3456           { l__enumext_leftmargin_tmp_#1_bool }
3457         \clist_map_inline:nn
3458           { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3459           { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
3460         \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3461           { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
3462         \skip_set_eq:Nc \parsep  { l__enumext_itemsep_#1_skip }
3463         \skip_zero:N \partopsep
3464         \usecounter { enumX#1 }
3465         \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
3466         \__enumext_starred_ref:
3467         \str_if_eq:nnTF {#1} { vii }
3468           {
3469             \__enumext_fake_item_vii:
3470             \bool_if:cT { l__enumext_show_length_vii_bool }
3471               { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3472           }
3473           {
3474             \__enumext_fake_item_viii:
3475             \bool_if:cT { l__enumext_show_length_#1_bool }
3476               { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3477           }
3478       }
3479   }
3480 \clist_map_inline:nn { vii, viii } { \__enumext_tmp:n {#1} }
```

(*End of definition for* __enumext_list_arg_two_vii: *and* __enumext_list_arg_two_viii:.)

12.38 The environment enumext

__enumext_safe_exec:

The __enumext_safe_exec: function first call the function __enumext_internal_mini_page: to create the environment __enumext_mini_page, then the function __enumext_is_not_nested: which sets \g__enumext_standar_bool to "*true*" if we are not nested within enumext*, we will increment \l__enumext_level_int to restrict nesting of the environment, set \l__enumext_standar_bool to "*true*" and finally call the function __enumext_is_on_first_level: which sets \l__enumext_standar_first_bool to "*true*" only if the environment is not nested and we are at the "*first level*".

```
3481 \cs_new_protected:Nn \__enumext_safe_exec:
3482   {
3483     \__enumext_internal_mini_page:
3484     \__enumext_is_not_nested:
3485     \int_incr:N \l__enumext_level_int
3486     \int_compare:nNnT { \l__enumext_level_int } > { 4 }
3487       { \msg_fatal:nn { enumext } { list-too-deep } }
3488     \bool_set_true:N \l__enumext_standar_bool
3489     \bool_set_false:N \l__enumext_starred_bool
3490     \__enumext_is_on_first_level:
3491   }
```

(*End of definition for* __enumext_safe_exec:.)

__enumext_parse_keys:n

The __enumext_parse_store_keys:n function first we will clear the variable \l__enumext_series_str used by the key series and then we check if we are at the "*first level*", if so we process the ⟨*keys*⟩ and then execute the function __enumext_parse_series:n used by the key series and call the function __enumext_nested_base_line_fix: used by the key base-fix, otherwise we will pass the ⟨*keys*⟩ to

the inner levels of the environment then we execute the function __enumext_store_active_keys:n and reprocess the ⟨keys⟩ to pass them to the storage ⟨sequence⟩ if the key save-key is not active.

```
3492  \cs_new_protected:Npn \__enumext_parse_keys:n #1
3493    {
3494      \tl_if_novalue:nF {#1}
3495        {
3496          \str_clear:N \l__enumext_series_str
3497          \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
3498            {
3499              \keys_set:nn { enumext / level-1 } {#1}
3500              \__enumext_parse_series:n {#1}
3501              \__enumext_nested_base_line_fix:
3502            }
3503            {
3504              \exp_args:Ne \keys_set:nn
3505                { enumext / level-\int_use:N \l__enumext_level_int } {#1}
3506            }
3507          \__enumext_store_active_keys:n {#1}
3508        }
3509    }
```

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

__enumext_start_store_level: The __enumext_start_store_level: function activate the level saving mechanism for *storage* in ⟨sequence⟩ for the command \anskey and the environment anskey*.

```
3510  \cs_new_protected:Nn \__enumext_start_store_level:
3511    {
3512      \bool_lazy_all:nT
3513        {
3514          { \bool_if_p:N \l__enumext_store_active_bool }
3515          { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3516          { \bool_if_p:N \g__enumext_standar_bool }
3517        }
3518        {
3519          \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3520            {
3521              \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3522              \__enumext_store_level_open:
3523            }
3524        }
```

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

```
3525      \bool_lazy_all:nT
3526        {
3527          { \bool_if_p:N \l__enumext_store_active_bool }
3528          { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3529          { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
3530        }
3531        {
3532          \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3533            {
3534              \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3535              \__enumext_store_level_open:
3536            }
3537        }
3538    }
```

(*End of definition for* __enumext_start_store_level:.)

__enumext_stop_store_level: The __enumext_stop_store_level: function stop the level saving mechanism for *storage* in ⟨sequence⟩ for the command \anskey and the environment anskey*.

```
3539  \cs_new_protected:Nn \__enumext_stop_store_level:
3540    {
3541      \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
3542        {
3543          \__enumext_store_level_close:
3544        }
3545    }
```

(*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.

```
3546 \cs_new_protected:Nn \__enumext_multicols_start:
3547   {
3548     \int_compare:nNnT
3549       { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3550       {
3551         \dim_compare:nNnT
3552           { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3553           {
3554             \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
3555               {
3556                 ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
3557                   + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
3558                 ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
3559                 - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3560               }
3561           }
3562         \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim }
3563         \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3564           {
3565             \dim_zero:N \columnseprule
3566           }
```

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.

```
3567         \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3568           {
3569             \skip_zero:N \multicolsep
3570             \__enumext_multi_addvspace:
3571           }
3572         \raggedcolumns
3573         \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
3574       }
3575   }
```

(*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:`.

```
3576 \cs_new_protected:Nn \__enumext_multicols_stop:
3577   {
3578     \int_compare:nNnTF
3579       { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3580       {
3581         \__enumext_stop_list:
3582         \__enumext_stop_store_level:
3583         \end{multicols}
3584         \__enumext_unskip_unkern:
3585         \__enumext_unskip_unkern:
3586         \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
3587       }
3588       {
3589         \__enumext_stop_list:
3590         \__enumext_stop_store_level:
3591       }
3592   }
```

(*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.

```
3593 \cs_new_protected:Nn \__enumext_before_list:
3594   {
3595     \__enumext_vspace_above:
3596     \__enumext_before_args_exec:
3597     \__enumext_check_ans_active:
```

When the `mini-env` key is active it will set the value of the `\l__enumext_minipage_right_X_dim` to be the *width* of the `__enumext_mini_page` environment on the *"right side"*, using this value together with the value of the `\l__enumext_minipage_hsep_X_dim` set by the `mini-sep` key, the value of `\l__enumext_minipage_left_X_dim` will be set, which will be the *width* of `__enumext_mini_page` environment on the *"left side"*, always having a current `\linewidth` as *maximum width* between them.

```
3598      \dim_compare:nNnT
3599        { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3600        {
3601          \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
3602            {
3603              \linewidth
3604              - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
3605              - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
3606            }
```

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.

```
3607          \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
3608          \int_gincr:N \g__enumext_minipage_stat_int
3609          \__enumext_minipage_add_space:
3610          \noindent
3611          \__enumext_mini_page{ \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
3612        }
3613      \__enumext_multicols_start:
3614    }
```

(*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.

```
3615  \cs_new_protected:Nn \__enumext_second_part:
3616    {
3617      \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
3618        {
3619          \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3620            {
3621              \msg_warning:nn { enumext } { missing-miniright }
3622              \miniright
3623            }
3624          \int_gzero:N \g__enumext_minipage_stat_int
3625          \__enumext_unskip_unkern: % remove topsep + [partopsep]
3626          \end__enumext_mini_page
3627        }
3628        {
3629          \__enumext_multicols_stop:
3630        }
```

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.

```
3631      \__enumext_after_stop_list:
3632      \__enumext_check_ans_key_hook:
3633      \__enumext_vspace_below:
3634      \bool_set_false:N \l__enumext_standar_bool
3635      \__enumext_resume_save_counter:
3636    }
```

(*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.

```
3637  \cs_new_protected:Nn \__enumext_set_item_width:
```

```
3638    {
3639      \dim_set:Nn \itemwidth { \linewidth }
3640      \dim_compare:nT
3641        {
3642          \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim } != \c_zero_dim
3643        }
3644        {
3645          \dim_sub:Nn \itemwidth
3646            {
3647              \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3648            }
3649        }
3650    }
```

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

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

```
3651  \NewDocumentEnvironment{enumext}{ O{} }
3652    {
3653      \__enumext_safe_exec:
3654      \__enumext_parse_keys:n {#1}
3655      \__enumext_before_list:
3656      \__enumext_start_store_level:
3657      \__enumext_start_list:nn
3658        { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
3659        {
3660          \use:c { __enumext_list_arg_two_ \__enumext_level: : }
3661          \__enumext_before_keys_exec:
3662        }
3663      \__enumext_set_item_width:
3664      \__enumext_after_args_exec:
3665    }
3666    {
3667      \__enumext_second_part:
3668    }
```

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.

```
3669  \__enumext_after_env:nn {enumext}
3670    {
3671      \__enumext_execute_after_env:
3672    }
```

(*End of definition for* enumext*. This function is documented on page* 5*.*)

12.39   The environment keyans

The environment keyans also based on lists. The main differences with the enumext environment are the *nesting* and the way the *answers* (choice) will be stored and checked, this environment is intended exclusively for *"multiple choice questions"*.

__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.

```
3673  \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3674    {
3675      \bool_if:NF \l__enumext_store_active_bool
3676        {
3677          \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
3678        }
3679      \int_incr:N \l__enumext_keyans_level_int
3680      \bool_set_true:N \l__enumext_keyans_env_bool
3681      \__enumext_keyans_name_and_start:
3682      % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3683      \bool_set_false:N \l__enumext_store_active_bool
3684      \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
3685        {
3686          \msg_error:nn { enumext } { keyans-nested }
3687        }
3688      \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3689        {
3690          \msg_error:nn { enumext } { keyans-wrong-level }
```

```
3691                }
3692            }
```

(*End of definition for* __enumext_keyans_safe_exec:.)

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

```
3693  \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3694    {
3695      \keys_set:nn { enumext / keyans } {#1}
3696    }
```

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

__enumext_before_list_v: Same implementation as the one used in the enumext environment.
__enumext_keyans_multicols_start:
__enumext_keyans_multicols_stop:
__enumext_second_part_v:

```
3697  \cs_new_protected:Nn \__enumext_before_list_v:
3698    {
3699      \__enumext_vspace_above_v:
3700      \__enumext_before_args_exec_v:
3701      \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3702        {
3703          \dim_set:Nn \l__enumext_minipage_left_v_dim
3704            {
3705              \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3706            }
3707          \bool_set_true:N \l__enumext_minipage_active_v_bool
3708          \int_gincr:N \g__enumext_minipage_stat_int
3709          \__enumext_keyans_minipage_add_space:
3710          \__enumext_mini_page{ \l__enumext_minipage_left_v_dim }
3711        }
3712      \__enumext_keyans_multicols_start:
3713    }
3714  \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3715    {
3716      \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3717        {
3718          \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3719            {
3720              \dim_set:Nn \l__enumext_columns_sep_v_dim
3721                {
3722                  (
3723                    \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3724                  ) / \l__enumext_columns_v_int
3725                  - \l__enumext_listoffset_v_dim
3726                }
3727            }
3728          \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim
3729          \dim_zero:N \columnseprule % no rule here
3730          \bool_if:NF \l__enumext_minipage_active_v_bool
3731            {
3732              \skip_zero:N \multicolsep
3733              \__enumext_keyans_multi_addvspace:
3734            }
3735          \raggedcolumns
3736          \begin{multicols}{ \l__enumext_columns_v_int }
3737        }
3738    }
3739  \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3740    {
3741      \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
3742        {
3743          \__enumext_stop_list:
3744          \end{multicols}
3745          \__enumext_unskip_unkern:
3746          \__enumext_unskip_unkern:
3747          \par\addvspace{ \l__enumext_multicols_below_v_skip }
3748        }
3749        {
3750          \__enumext_stop_list:
3751        }
3752    }
3753  \cs_new_protected:Nn \__enumext_second_part_v:
```

```
3754  {
3755    \bool_if:NTF \l__enumext_minipage_active_v_bool
3756      {
3757        \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3758          {
3759            \msg_warning:nn { enumext } { missing-miniright }
3760            \miniright
3761          }
3762        \int_gzero:N \g__enumext_minipage_stat_int
3763        \__enumext_unskip_unkern: % remove \topsep + [\partopsep]
3764        \end__enumext_mini_page
3765        \par\addvspace{ \l__enumext_minipage_after_skip }
3766      }
3767      {
3768        \__enumext_keyans_multicols_stop:
3769      }
3770    \bool_set_false:N \l__enumext_keyans_env_bool
3771    \__enumext_after_stop_list_v:
3772    \__enumext_vspace_below_v:
3773  }
```

(*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.

```
3774  \cs_new_protected:Nn \__enumext_keyans_set_item_width:
3775  {
3776    \dim_set:Nn \itemwidth { \linewidth }
3777    \dim_compare:nT
3778      {
3779        \l__enumext_listoffset_v_dim != \c_zero_dim
3780      }
3781      {
3782        \dim_sub:Nn \itemwidth { \l__enumext_listoffset_v_dim }
3783      }
3784  }
```

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

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

```
3785  \NewDocumentEnvironment{keyans}{ O{} }
3786  {
3787    \__enumext_keyans_safe_exec:
3788    \__enumext_keyans_parse_keys:n {#1}
3789    \__enumext_before_list_v:
3790    \__enumext_start_list:nn
3791      { \tl_use:N \l__enumext_label_v_tl }
3792      {
3793        \__enumext_list_arg_two_v:
3794        \__enumext_before_keys_exec_v:
3795      }
3796    \__enumext_keyans_set_item_width:
3797    \__enumext_after_args_exec_v:
3798  }
3799  {
3800    \__enumext_check_starred_cmd:n { item }
3801    \__enumext_second_part_v:
3802  }
```

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

12.40   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. This implementation is an adaptation of my answer thanks to Ulrike Fischer's comments in How can I modify my `\item` redefinition to be compatible with `tagging-pdf`.

12.40.1 Socket for tagging support in `enumext*` and `keyans*`

start-list-tags
stop-start-tags
stop-list-tags
__enumext_start_list_tag:n
__enumext_stop_start_list_tag:
__enumext_stop_list_tag:n

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

```
3803 \socket_new:nn {tagsupport/enumext/starred}{ 1 }
3804 \socket_new_plug:nnn {tagsupport/enumext/starred} {start-list-tags}
3805   {
3806     \tag_resume:n {#1}
3807     \tag_struct_begin:n {tag=LI}
3808     \tag_struct_begin:n {tag=Lbl}
3809     \tag_mc_begin:n {tag=Lbl}
3810   }
3811 \socket_new_plug:nnn {tagsupport/enumext/starred} {stop-start-tags}
3812   {
3813     \tag_mc_end:
3814     \tag_struct_end:n {tag=Lbl}
3815     \tag_struct_begin:n {tag=LBody}
3816     \tag_struct_begin:n {tag=text-unit}
3817     \tag_struct_begin:n {tag=text}
3818   }
3819 \socket_new_plug:nnn {tagsupport/enumext/starred} {stop-list-tags}
3820   {
3821     \tag_struct_end:n {tag=text}
3822     \tag_struct_end:n {tag=text-unit}
3823     \tag_struct_end:n {tag=LBody}
3824     \tag_struct_end:n {tag=LI}
3825     \tag_suspend:n {#1}
3826   }
```

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

```
3827 \cs_new_protected_nopar:Npn \__enumext_start_list_tag:n #1
3828   {
3829     \IfDocumentMetadataTF
3830       {
3831         \socket_assign_plug:nn {tagsupport/enumext/starred} {start-list-tags}
3832         \socket_use:n {tagsupport/enumext/starred} {#1}
3833       } {}
3834   }
3835 \cs_new_protected_nopar:Nn \__enumext_stop_start_list_tag:
3836   {
3837     \IfDocumentMetadataTF
3838       {
3839         \socket_assign_plug:nn {tagsupport/enumext/starred} {stop-start-tags}
3840         \socket_use:nn {tagsupport/enumext/starred} { }
3841       } {}
3842   }
3843 \cs_new_protected_nopar:Npn \__enumext_stop_list_tag:n #1
3844   {
3845     \IfDocumentMetadataTF
3846       {
3847         \socket_assign_plug:nn {tagsupport/enumext/starred} {stop-list-tags}
3848         \socket_use:nn {tagsupport/enumext/starred} {#1}
3849       } {}
3850   }
```

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

12.40.2 Socket for tagging support in `keyanspic`

start-list-tags
stop-start-tags
stop-list-tags
__enumext_anspic_start_list_tag:
__enumext_anspic_stop_start_list_tag:
__enumext_anspic_stop_list_tag:

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

```
3851 \socket_new:nn {tagsupport/enumext/keyanspic}{ 0 }
3852 \socket_new_plug:nnn {tagsupport/enumext/keyanspic} {start-list-tags}
3853   {
3854     \tag_resume:n {keyanspic}
3855     \tag_struct_begin:n {tag=LI}
3856     \tag_struct_begin:n {tag=Lbl}
3857     \tag_mc_begin:n {tag=Lbl}
3858   }
3859 \socket_new_plug:nnn {tagsupport/enumext/keyanspic} {stop-start-tags}
3860   {
3861     \tag_mc_end:
3862     \tag_struct_end:n {tag=Lbl}
3863     \tag_struct_begin:n {tag=LBody}
3864     \tag_struct_begin:n {tag=text-unit}
```

```
3865        \tag_struct_begin:n {tag=text}
3866        \tag_mc_begin:n {tag=text}
3867    }
3868 \socket_new_plug:nnn {tagsupport/enumext/keyanspic} {stop-list-tags}
3869    {
3870      \tag_mc_end:
3871      \tag_struct_end:n {tag=text-unit}
3872      \tag_struct_end:n {tag=text}
3873      \tag_struct_end:n {tag=LBody}
3874      \tag_struct_end:n {tag=LI}
3875      \tag_suspend:n {keyanspic}
3876    }
```

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

```
3877 \cs_new_protected_nopar:Nn \__enumext_anspic_start_list_tag:
3878    {
3879      \IfDocumentMetadataTF
3880        {
3881          \socket_assign_plug:nn {tagsupport/enumext/keyanspic} {start-list-tags}
3882          \socket_use:n {tagsupport/enumext/keyanspic}
3883        } {}
3884    }
3885 \cs_new_protected_nopar:Nn \__enumext_anspic_stop_start_list_tag:
3886    {
3887      \IfDocumentMetadataTF
3888        {
3889          \socket_assign_plug:nn {tagsupport/enumext/keyanspic} {stop-start-tags}
3890          \socket_use:nn {tagsupport/enumext/keyanspic}
3891        } {}
3892    }
3893 \cs_new_protected_nopar:Nn \__enumext_anspic_stop_list_tag:
3894    {
3895      \IfDocumentMetadataTF
3896        {
3897          \socket_assign_plug:nn {tagsupport/enumext/keyanspic} {stop-list-tags}
3898          \socket_use:nn {tagsupport/enumext/keyanspic}
3899        } {}
3900    }
```

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

12.41 The environment keyanspic and \anspic

The `keyanspic` environment is a `list` based environment that uses the same configuration for *"spacing"* and ⟨*label*⟩ as the `keyans` environment, but it does not use `\item`. The ⟨*contents*⟩ are passed to the environment by means of the `\anspic` command 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.
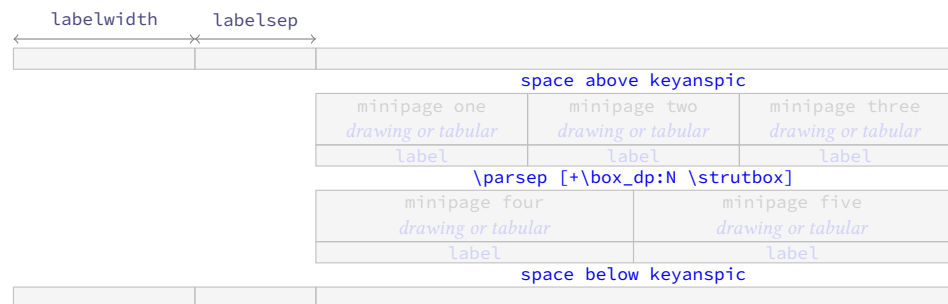
Figure 12: Representation of the `keyanspic` spacing in enumext.

The environment `keyanspic` will take two arguments, the first *starred argument* '`*`' will set the position of the ⟨*label*⟩ processed by the command `\anspic` which will be *"above"* if present and *"below"* otherwise, the second *optional argument* 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 environment, if not present these will be printed on a *single line.*

🔖 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.

12.41.1 The environment keyanspic

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?.

__enumext_keyans_pic_safe_exec:n

The function __enumext_keyans_pic_safe_exec:n check the *starred argument* '*' and nested level position inside the enumext environment. We will set the state of the variable \l__enumext_keyans_pic_-star_bool along with the value of the variable \l__enumext_anspic_mini_pos_str using by \anspic according to the presence of the *starred argument* '*'.

```
3901 \cs_new_protected:Npn \__enumext_keyans_pic_safe_exec:n #1
3902   {
3903     \int_incr:N \l__enumext_keyans_pic_level_int
3904     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
3905       {
3906         \msg_error:nn { enumext } { keyanspic-nested }
3907       }
3908     \__enumext_keyans_name_and_start:
3909     \bool_if:nTF { #1 }
3910       {
3911         \bool_set_true:N \l__enumext_keyans_pic_star_bool
3912         \str_set:Nn \l__enumext_anspic_mini_pos_str { t }
3913       }
3914       {
3915         \str_set:Nn \l__enumext_anspic_mini_pos_str { b }
3916       }
3917   }
```

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

__enumext_keyans_pic_skip_abs:N

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

```
3918 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
3919   {
3920     \dim_compare:nNnT { #1 } < { \c_zero_dim }
3921       {
3922         \skip_set:Nn #1 { -#1 }
3923       }
3924   }
```

(*End of definition for* __enumext_keyans_pic_skip_abs:N.)

__enumext_keyans_pic_arg_two:

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 and wrap-label 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.

```
3925 \cs_new_protected:Nn \__enumext_keyans_pic_arg_two:
3926   {
3927     \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
3928     \__enumext_list_arg_two_v:
3929     \__enumext_keyans_pic_skip_abs:N \parsep
```

Now we increment the enumXv counter 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 *starred argument* '*' is NOT present.

```
3930     \bool_if:NF \l__enumext_keyans_pic_star_bool
3931       {
3932         \stepcounter { enumXv }
3933         \hbox_set:Nn \l__enumext_anspic_label_box { \l__enumext_label_v_tl }
3934         \dim_set:Nn \l__enumext_anspic_label_htdp_dim
3935           {
3936             \box_ht_plus_dp:N \l__enumext_anspic_label_box
3937           }
3938         \skip_add:Nn \parsep
3939           {
3940             \l__enumext_anspic_label_htdp_dim + \box_dp:N \strutbox
3941           }
3942         \skip_gset_eq:NN \g__enumext_keyans_pic_parsep_skip \parsep
3943       }
```

Finally we adjust the value of `\leftmargin` and `\topsep` then set `\labelwidth`, `\labelsep`, `\partopsep` and `\itemsep` to zero so that the *horizontal* and *vertical* space is not affected.

```
3944    \dim_add:Nn  \leftmargin { -\labelwidth - \labelsep }
3945    \skip_add:Nn \topsep { 0.5\box_dp:N \strutbox }
3946    \dim_zero:N  \labelwidth
3947    \dim_zero:N  \listparindent
3948    \dim_zero:N  \labelsep
3949    \skip_zero:N \partopsep
3950    \skip_zero:N \itemsep
3951  }
```

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

keyanspic   Now we define the environment `keyanspic`. For compatibility with *tagged* PDF we must use the `\beginlist` form and a lot of conditional code using `\IfDocumentMetadataTF`.

```
3952  \NewDocumentEnvironment{keyanspic}{ s o }
3953    {
3954      \__enumext_keyans_pic_safe_exec:n { #1 }
3955      \begin{list} { } { \__enumext_keyans_pic_arg_two: }
3956      \IfDocumentMetadataTF
3957        {
3958          \tag_suspend:n {list}
3959        }{}
3960      \item[] \scan_stop:
3961      % paranoia
3962      \RenewDocumentCommand \item {}
3963        {
3964          \msg_error:nn { enumext } { keyanspic-item-cmd }
3965        }
3966      \IfDocumentMetadataTF
3967        {
3968          \tag_resume:n {keyanspic}
3969          \tag_tool:n {para/tagging=false}
3970          \tag_suspend:n {keyanspic}
3971        } { }
3972    }
3973    {
3974      \IfDocumentMetadataTF
3975        {
3976          \tag_resume:n {keyanspic}
3977          \tag_struct_begin:n {tag=L,attribute=enumerate}
3978        } { }
```

Now we process the command `\anspic`, if the *optional argument* is not present, the number of times the `\anspic` command appears will be counted from `\l__enumext_anspic_args_seq` and placed a single line.

```
3979      \tl_if_novalue:nTF { #2 }
3980        {
3981          \__enumext_anspic_print:e { \seq_count:N \l__enumext_anspic_args_seq }
3982        }
3983        { \__enumext_anspic_print:n { #2 } }
3984      \IfDocumentMetadataTF
3985        {
3986          \tag_suspend:n {keyanspic}
3987        } { }
3988      \end{list}
3989      \IfDocumentMetadataTF
3990        {
3991          \tag_struct_end:
3992          \tag_struct_end:
3993        } { }
```

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

```
3994      \__enumext_check_starred_cmd:n { anspic }
3995      \setcounter { enumXvi } { 0 }
3996      \bool_if:NTF \l__enumext_keyans_pic_star_bool
3997        {
3998          \par\addvspace{ 0.5\box_dp:N \strutbox }
3999        }
4000        {
4001          \par\addvspace{ \g__enumext_keyans_pic_parsep_skip }
```

```
4002            }
4003     %\bool_set_false:N \l__enumext_store_active_bool
4004    }
```

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

12.41.2 The command \anspic

The \anspic command take three arguments, the *starred versions* \anspic*[⟨content⟩] *store* the current ⟨*label*⟩ next to the [⟨content⟩] (if it is present) in the ⟨*sequence*⟩ and ⟨*prop list*⟩ defined by save-ans key. The third (mandatory) argument *"drawing or tabular"* is NOT *stored* in the ⟨*sequence*⟩ or ⟨*prop list*⟩.

\anspic 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.

```
4005 \NewDocumentCommand \anspic { s o +m }
4006   {
4007     \bool_if:NF \l__enumext_store_active_bool
4008       {
4009         \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
4010       }
4011     \int_compare:nNnT { \l__enumext_level_int } > { 1 }
4012       {
4013         \msg_error:nn { enumext } { keyanspic-wrong-level }
4014       }
4015     \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
4016       {
4017         \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
4018       }
4019     \seq_put_right:Nn \l__enumext_anspic_args_seq
4020       {
4021         \__enumext_anspic_args:nnn { #1 } { #2 } { #3 }
4022       }
4023   }
```

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

__enumext_anspic_body_dim:n The __enumext_anspic_body_dim:n function will set the value of \l__enumext_anspic_body_htdp_-dim equal to the height and depth of the mandatory argument if the keyanspic* environment is used with the *starred argument* '*'.

```
4024 \cs_new_protected:Npn \__enumext_anspic_body_dim:n #1
4025   {
4026     \bool_if:NF \l__enumext_keyans_pic_star_bool
4027       {
4028         \IfDocumentMetadataTF
4029           {
4030             \tag_suspend:n {keyanspic}
4031           } { }
4032         \vbox_set:Nn \l__enumext_anspic_body_box { #1 }
4033         \dim_set:Nn \l__enumext_anspic_body_htdp_dim
4034           {
4035             \box_ht_plus_dp:N \l__enumext_anspic_body_box
4036           }
4037         \IfDocumentMetadataTF
4038           {
4039             \tag_resume:n {keyanspic}
4040           } { }
4041       }
4042   }
```

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

__enumext_anspic_label:nn 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 and wrap-opt keys.

```
4043 \cs_new_protected:Npn \__enumext_anspic_label:nn #1 #2
4044   {
4045     \makebox[ \l__enumext_anspic_mini_width_dim ][ c ]
4046       {
4047         \bool_if:nT { #1 }
```

```
4048                { 
4049                    \__enumext_keyans_addto_prop:n { #2 }
4050                    \__enumext_keyans_store_ref:
4051                    \__enumext_keyans_addto_seq:n { #2 }
4052                    \int_gincr:N \g__enumext_check_starred_cmd_int
4053                    \bool_lazy_or:nnT
4054                      { \bool_if_p:N \l__enumext_show_answer_bool }
4055                      { \bool_if_p:N \l__enumext_show_position_bool }
4056                      {
4057                        \tl_set_eq:NN \l__enumext_label_v_tl \l__enumext_label_vi_tl
4058                        \__enumext_keyans_show_left:n { #2 }
4059                        \tl_set_eq:NN \l__enumext_label_vi_tl \l__enumext_label_v_tl
4060                      }
4061                  }
4062                \tl_use:N \l__enumext_label_font_style_v_tl
4063                \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl }
4064                \__enumext_keyans_show_item_opt:
4065              }
4066          }
```

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

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

```
4067  \cs_new_protected:Npn \__enumext_anspic_label_pos:nnn #1 #2 #3
4068    {
4069      \stepcounter { enumXvi }
4070      \__enumext_anspic_body_dim:n { #3 }
4071      \bool_if:NTF \l__enumext_keyans_pic_star_bool
4072        {
4073          \__enumext_anspic_label:nn { #1 } { #2 }
4074        }
4075        {
4076          \raisebox
4077            {
4078              -\dim_eval:n
4079                {
4080                  \l__enumext_anspic_label_htdp_dim
4081                  + \l__enumext_anspic_body_htdp_dim
4082                  + \box_dp:N \strutbox
4083                }
4084            }
4085            [ 0pt ] [ 0pt ]
4086            {
4087              \__enumext_anspic_label:nn { #1 } { #2 }
4088            }
4089        }
4090    }
4091  %
```

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

`\__enumext_anspic_args:nnn`   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.

```
4092  \cs_new_protected:Nn \__enumext_anspic_args:nnn
4093    {
4094      \__enumext_anspic_start_list_tag:
4095      \__enumext_anspic_label_pos:nnn { #1 } { #2 } { #3 }
4096      \__enumext_anspic_stop_start_list_tag:
4097      \\ #3
4098      \__enumext_anspic_stop_list_tag:
4099    }
```

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

`\__enumext_anspic_print:n`
`\__enumext_anspic_print:e`
`\__enumext_anspic_row:n`
The *optional argument* [⟨*n° upper, n° lower*⟩] passed to the keyanspic environment is split by comma and is handled directly by the function `\__enumext_anspic_print:n` and passed to the function `\__enumext_-anspic_row:n`.

```
4100  \cs_new_protected:Nn \__enumext_anspic_print:n
4101    {
```

```
4102        \clist_map_function:nN { #1 } \__enumext_anspic_row:n
4103    }
4104  \cs_generate_variant:Nn \__enumext_anspic_print:n { e }
```

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.

```
4105  \cs_new_protected:Nn \__enumext_anspic_row:n
4106    {
4107      \dim_set:Nn \l__enumext_anspic_mini_width_dim { \linewidth / #1 }
4108      \int_set:Nn \l__enumext_anspic_above_int { \l__enumext_anspic_below_int }
4109      \int_set:Nn \l__enumext_anspic_below_int { \l__enumext_anspic_above_int + #1 }
4110      \int_step_inline:nnn
4111        { \l__enumext_anspic_above_int + 1 }
4112        { \l__enumext_anspic_below_int }
4113        {
4114          \IfDocumentMetadataTF
4115            {
4116              \tag_suspend:n {minipage}
4117            } { }
4118          \begin{minipage}[ \l__enumext_anspic_mini_pos_str ]{ \l__enumext_anspic_mini_width_dim }
4119            \centering
4120            \seq_item:Nn \l__enumext_anspic_args_seq { ##1 }
4121          \end{minipage}
4122          \IfDocumentMetadataTF
4123            {
4124              \tag_resume:n {minipage}
4125            } { }
4126        }
4127      \par
4128    }
```

(*End of definition for __enumext_anspic_print:n and __enumext_anspic_row:n.*)

12.42   The horizontal environments

Generating *horizontal list environments* is NOT as simple as standard LaTeX list environments. The fundamental part of the code is adapted from the shortlst package to a more modern version using expl3. It is not possible to redefine \item and \makelabel 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.

12.42.1   Functions for item box width

We set the default value for the *width of the box* containing the ⟨*content*⟩ of the items for enumext* environment.

__enumext_starred_columns_set_vii:
__enumext_starred_columns_set_viii:

```
4129  \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
4130    {
4131      \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
4132        {
4133          \dim_set:Nn \l__enumext_columns_sep_vii_dim
4134            {
4135              ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
4136              / \l__enumext_columns_vii_int
4137            }
4138        }
4139      \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - 1 }
4140      \dim_set:Nn \l__enumext_item_width_vii_dim
```

```
4141        {
4142            ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
4143            / \l__enumext_columns_vii_int
4144            - \l__enumext_labelwidth_vii_dim
4145            - \l__enumext_labelsep_vii_dim
4146        }
```

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

```
4147        \dim_compare:nNnT { \l__enumext_rightmargin_vii_dim } > { \c_zero_dim }
4148        {
4149            \dim_sub:Nn \l__enumext_item_width_vii_dim
4150            {
4151                ( \l__enumext_rightmargin_vii_dim * \l__enumext_tmpa_vii_int )
4152                / \l__enumext_columns_vii_int
4153            }
4154            \dim_add:Nn \l__enumext_columns_sep_vii_dim
4155            {
4156                \l__enumext_rightmargin_vii_dim
4157            }
4158        }
4159    }
```

Same implementation for the `keyans*` environment.

```
4160    \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
4161    {
4162        \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
4163        {
4164            \dim_set:Nn \l__enumext_columns_sep_viii_dim
4165            {
4166                ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
4167                / \l__enumext_columns_viii_int
4168            }
4169        }
4170        \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - 1 }
4171        \dim_set:Nn \l__enumext_item_width_viii_dim
4172        {
4173            ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
4174            / \l__enumext_columns_viii_int
4175            - \l__enumext_labelwidth_viii_dim
4176            - \l__enumext_labelsep_viii_dim
4177        }
4178        \dim_compare:nNnT { \l__enumext_rightmargin_viii_dim } > { \c_zero_dim }
4179        {
4180            \dim_sub:Nn \l__enumext_item_width_viii_dim
4181            {
4182                ( \l__enumext_rightmargin_viii_dim * \l__enumext_tmpa_vii_int )
4183                / \l__enumext_columns_viii_int
4184            }
4185            \dim_add:Nn \l__enumext_columns_sep_viii_dim
4186            {
4187                \l__enumext_rightmargin_viii_dim
4188            }
4189        }
4190    }
```

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

12.42.2   Functions for join item columns

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

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

```
4191    \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
4192    {
4193        \int_set:Nn \l__enumext_joined_item_vii_int {#1}
4194        \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
4195        {
4196            \msg_warning:nnee { enumext } { item-joined }
4197                { \int_use:N \l__enumext_joined_item_vii_int }
4198                { \int_use:N \l__enumext_columns_vii_int }
4199            \int_set:Nn \l__enumext_joined_item_vii_int
4200            {
4201                \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
```

```
4202                    }
4203                 }
4204          \int_compare:nNnT
4205             { \l__enumext_joined_item_vii_int }
4206             >
4207             { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4208             {
4209                \msg_warning:nnee { enumext } { item-joined-columns }
4210                  { \int_use:N \l__enumext_joined_item_vii_int }
4211                  {
4212                     \int_eval:n
4213                       { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4214                  }
4215                \int_set:Nn \l__enumext_joined_item_vii_int
4216                  {
4217                     \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4218                  }
4219             }
4220          \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { 1 }
4221             {
4222                \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
4223                \int_decr:N \l__enumext_joined_item_aux_vii_int
4224                \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
4225                \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
4226                \dim_set:Nn \l__enumext_joined_width_vii_dim
4227                  {
4228                     \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
4229                     + (  \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
4230                         + \l__enumext_columns_sep_vii_dim
4231                       )*\l__enumext_joined_item_aux_vii_int
4232                  }
4233                \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
4234             }
4235             {
4236                \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
4237                \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
4238             }
4239       }
```

Same implementation for the keyans* environment.

```
4240   \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
4241     {
4242        \int_set:Nn \l__enumext_joined_item_viii_int {#1}
4243        \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
4244           {
4245              \msg_warning:nnee { enumext } { item-joined }
4246                { \int_use:N \l__enumext_joined_item_viii_int }
4247                { \int_use:N \l__enumext_columns_viii_int }
4248              \int_set:Nn \l__enumext_joined_item_viii_int
4249                {
4250                   \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4251                }
4252           }
4253        \int_compare:nNnT
4254           { \l__enumext_joined_item_viii_int }
4255           >
4256           { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4257           {
4258              \msg_warning:nnee { enumext } { item-joined-columns }
4259                { \int_use:N \l__enumext_joined_item_viii_int }
4260                {
4261                   \int_eval:n
4262                     { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4263                }
4264              \int_set:Nn \l__enumext_joined_item_viii_int
4265                {
4266                   \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4267                }
4268           }
4269        \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { 1 }
4270           {
4271              \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
```

```
4272        \int_decr:N \l__enumext_joined_item_aux_viii_int
4273        \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
4274        \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
4275        \dim_set:Nn \l__enumext_joined_width_viii_dim
4276          {
4277            \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
4278            + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
4279              + \l__enumext_columns_sep_viii_dim
4280            )*\l__enumext_joined_item_aux_viii_int
4281          }
4282        \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
4283      }
4284      {
4285        \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
4286        \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
4287      }
4288    }
```

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

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

__enumext_start_mini_vii: The implementation of the mini-env key support is almost identical to the one used in the enumext and
__enumext_stop_mini_vii: 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.

```
4289  \cs_new_protected:Nn \__enumext_start_mini_vii:
4290    {
4291      \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
4292        {
4293          \dim_set:Nn \l__enumext_minipage_left_vii_dim
4294            {
4295              \linewidth
4296              - \l__enumext_minipage_right_vii_dim
4297              - \l__enumext_minipage_hsep_vii_dim
4298            }
4299          \bool_set_true:N \l__enumext_minipage_active_vii_bool
4300          \dim_gset_eq:NN
4301            \g__enumext_minipage_right_vii_dim
4302            \l__enumext_minipage_right_vii_dim
4303          \__enumext_mini_addvspace_vii:
4304          \nointerlineskip\noindent
4305          \__enumext_mini_page{ \l__enumext_minipage_left_vii_dim }
4306        }
4307    }
```

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 (§12.43).

```
4308  \cs_new_protected:Nn \__enumext_stop_mini_vii:
4309    {
4310      \bool_if:NTF \l__enumext_minipage_active_vii_bool
4311        {
4312          \__enumext_stop_list:
4313          \__enumext_stop_store_level_vii:
4314          \IfDocumentMetadataTF { \tag_resume:n {enumext*} } { }
4315          \end__enumext_mini_page
4316          \hfill
4317          \bool_gset_true:N \g__enumext_minipage_active_vii_bool
4318        }
4319        {
4320          \__enumext_stop_list:
4321          \__enumext_stop_store_level_vii:
4322        }
4323    }
```

(*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.

```
4324  \__enumext_after_env:nn {enumext*}
4325    {
4326      \bool_if:NT \g__enumext_minipage_active_vii_bool
4327        {
4328          \__enumext_minipage:w [ t ] { \g__enumext_minipage_right_vii_dim }
4329            \legacy_if_gset_false:n { @minipage }
4330            \skip_vertical:N \c_zero_skip
4331            \par\addvspace { \g__enumext_minipage_right_skip }
4332            \bool_if:NF \g__enumext_minipage_center_vii_bool
4333              {
4334                \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
4335                  {
4336                    \centering
4337                  }
4338              }
4339            \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
4340              {
4341                \tl_use:N \g__enumext_miniright_code_vii_tl
4342              }
4343            \box_use_drop:N \l__enumext_miniright_code_vii_box
4344            \skip_vertical:N \c_zero_skip
4345          \__enumext_endminipage:
4346          \par\addvspace{ \g__enumext_minipage_after_skip }
4347        }
4348      \bool_gset_false:N \g__enumext_minipage_active_vii_bool
4349      \bool_gset_true:N \g__enumext_minipage_center_vii_bool
4350      \tl_gclear:N \g__enumext_miniright_code_vii_tl
4351      \dim_gzero:N \g__enumext_minipage_right_vii_dim
4352      \bool_gset_false:N \g__enumext_starred_bool
4353    }
```

`\__enumext_start_mini_viii:`
`\__enumext_stop_mini_viii:`

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

```
4354  \cs_new_protected:Nn \__enumext_start_mini_viii:
4355    {
4356      \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
4357        {
4358          \dim_set:Nn \l__enumext_minipage_left_viii_dim
4359            {
4360              \linewidth
4361              - \l__enumext_minipage_right_viii_dim
4362              - \l__enumext_minipage_hsep_viii_dim
4363            }
4364          \bool_set_true:N \l__enumext_minipage_active_viii_bool
4365          \dim_gset_eq:NN
4366            \g__enumext_minipage_right_viii_dim
4367            \l__enumext_minipage_right_viii_dim
4368          \__enumext_mini_addvspace_viii:
4369          \nointerlineskip\noindent
4370          \__enumext_mini_page{ \l__enumext_minipage_left_viii_dim }
4371        }
4372    }
4373  \cs_new_protected:Nn \__enumext_stop_mini_viii:
4374    {
4375      \bool_if:NTF \l__enumext_minipage_active_viii_bool
4376        {
4377          \__enumext_stop_list:
4378          \IfDocumentMetadataTF { \tag_resume:n {keyans*} } { }
4379          \end__enumext_mini_page
4380          \hfill
4381          \bool_gset_true:N \g__enumext_minipage_active_viii_bool
4382        }
4383        {
4384          \__enumext_stop_list:
4385        }
4386    }
```

```
4387 \__enumext_after_env:nn {keyans*}
4388   {
4389     \bool_if:NT \g__enumext_minipage_active_viii_bool
4390       {
4391         \__enumext_mini_page{ \g__enumext_minipage_right_viii_dim }
4392         \par\addvspace { \g__enumext_minipage_right_skip }
4393         \bool_if:NF \g__enumext_minipage_center_viii_bool
4394           {
4395             \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
4396               {
4397                 \centering
4398               }
4399           }
4400         \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
4401           {
4402             \tl_use:N \g__enumext_miniright_code_viii_tl
4403           }
4404         \box_use_drop:N \l__enumext_miniright_code_viii_box
4405         \end__enumext_mini_page
4406         \par\addvspace{ \g__enumext_minipage_after_skip }
4407       }
4408     \bool_gset_false:N \g__enumext_minipage_active_viii_bool
4409     \bool_gset_true:N \g__enumext_minipage_center_viii_bool
4410     \tl_gclear:N \g__enumext_miniright_code_viii_tl
4411     \dim_gzero:N \g__enumext_minipage_right_viii_dim
4412   }
```

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

12.42.4 Redefining \footnote **command**

__enumext_footnotetext:nn
__enumext_renew_footnote:
__enumext_print_footnote:

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

```
4413 \cs_new_protected:Nn \__enumext_footnotetext:nn
4414   {
4415     \footnotetext[#1]{#2}
4416   }
4417 \cs_new_protected:Nn \__enumext_renew_footnote:
4418   {
4419     \seq_gclear:N \g__enumext_footnote_arg_seq
4420     \seq_gclear:N \g__enumext_footnote_int_seq
4421     \RenewDocumentCommand \footnote { o +m }
4422       {
4423         \tl_if_novalue:nTF {##1}
4424           {
4425             \stepcounter{footnote}
4426             \int_gset_eq:Nc \g__enumext_footnote_int { c@footnote }
4427           }
4428           {
4429             \int_gset:Nn \g__enumext_footnote_int { ##1 }
4430           }
4431         \footnotemark [ \g__enumext_footnote_int ]
4432         \seq_gput_right:Nn \g__enumext_footnote_arg_seq { ##2 }
4433         \seq_gput_right:NV \g__enumext_footnote_int_seq \g__enumext_footnote_int
4434       }
4435   }
4436 \cs_new_protected:Nn \__enumext_print_footnote:
4437   {
4438     \seq_if_empty:NF \g__enumext_footnote_int_seq
4439       {
4440         \seq_map_pairwise_function:NNN
4441           \g__enumext_footnote_int_seq
4442           \g__enumext_footnote_arg_seq
4443           \__enumext_footnotetext:nn
4444       }
4445   }
```

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

12.43 The environment enumext*

enumext* First we will generate the environment and we will give a temporary definition to __enumext_stop_-item_tmp_vii: equal to __enumext_first_item_tmp_vii: and next to \item equal to __enumext_-start_item_tmp_vii: which we will redefine later. Unlike the implementation used by the shortlst package, we will not set the values of \rightskip and \@rightskip equal to \@flushglue whose value is 0.0pt plus 1.0 fil, in the tests I have performed this fails in some circumstances and different results are obtained when using pdfTeX and LuaTeX.

```
4446  \NewDocumentEnvironment{enumext*}{ o }
4447    {
4448      \__enumext_safe_exec_vii:
4449      \__enumext_parse_keys_vii:n {#1}
4450      \__enumext_before_list_vii:
4451      \__enumext_start_store_level_vii:
4452      \__enumext_start_list:nn { }
4453        {
4454          \__enumext_list_arg_two_vii:
4455          \__enumext_before_keys_exec_vii:
4456        }
4457      % Stop tagging
4458      \IfDocumentMetadataTF { \tag_suspend:n {enumext*} } { }
4459      \__enumext_starred_columns_set_vii:
4460      \item[] \scan_stop:
4461      \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_first_item_tmp_vii:
4462      \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
4463      \ignorespaces
4464    }
4465    {
4466      \IfDocumentMetadataTF { \tag_struct_end:n {tag=text-unit} } { }
4467      \__enumext_stop_item_tmp_vii:
4468      \__enumext_remove_extra_parsep_vii:
4469      \__enumext_after_list_vii:
4470    }
```

(*End of definition for* enumext*. *This function is documented on page 5.*)

__enumext_safe_exec_vii: We will first call the function __enumext_internal_mini_page: to create the environment __enumext_-mini_page, then the function __enumext_is_not_nested: which sets \g__enumext_starred_bool to true if we are not nested within enumext, we will increment \l__enumext_level_h_int to restrict nesting of the environment, set \l__enumext_starred_bool to true and finally call the function __enumext_is_-on_first_level: which sets \l__enumext_starred_first_bool to true if we are not nested, allowing the *"storage system"* to be used.

```
4471  \cs_new_protected:Nn \__enumext_safe_exec_vii:
4472    {
4473      \__enumext_internal_mini_page:
4474      \__enumext_is_not_nested:
4475      \int_incr:N \l__enumext_level_h_int
4476      \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
4477        {
4478          \msg_error:nn { enumext } { nested }
4479        }
4480      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
4481        {
4482          \msg_error:nnn { enumext } { nested-horizontal } { keyans*}
4483        }
4484      \bool_set_true:N \l__enumext_starred_bool
4485      \bool_set_false:N \l__enumext_standar_bool
4486      \__enumext_is_on_first_level:
4487    }
```

(*End of definition for* __enumext_safe_exec_vii:.)

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

```
4488  \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
4489    {
4490      \tl_if_novalue:nF {#1}
```

```
4491        {
4492            \str_clear:N \l__enumext_series_str
4493            \keys_set:nn { enumext / enumext* } {#1}
4494            \__enumext_parse_series:n {#1}
4495            \__enumext_store_active_keys_vii:n {#1}
4496            \__enumext_nested_base_line_fix:
4497        }
4498    }
```

(*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*.

```
4499 \cs_new_protected:Nn \__enumext_before_list_vii:
4500    {
4501        \__enumext_vspace_above_vii:
4502        \__enumext_check_ans_active:
4503        \__enumext_before_args_exec_vii:
4504        \__enumext_start_mini_vii:
4505    }
```

(*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: (§12.42.3) 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.

```
4506 \cs_new_protected:Nn \__enumext_after_list_vii:
4507    {
4508        \__enumext_stop_mini_vii:
4509        \__enumext_after_stop_list_vii:
4510        \__enumext_check_ans_key_hook:
4511        \__enumext_vspace_below_vii:
4512        \bool_set_false:N \l__enumext_starred_bool
4513        \__enumext_resume_save_counter:
4514    }
```

(*End of definition for* __enumext_after_list_vii:.)

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

```
4515 \cs_new_protected:Nn \__enumext_start_store_level_vii:
4516    {
4517        \bool_if:NT \l__enumext_store_active_bool
4518            {
4519                \int_compare:nNnT { \l__enumext_level_int } > { 0 }
4520                    {
4521                        \__enumext_store_level_open_vii:
4522                    }
4523            }
4524    }
4525 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
4526    {
4527        \bool_if:NT \l__enumext_store_active_bool
4528            {
4529                \int_compare:nNnT { \l__enumext_level_int } > { 0 }
4530                    {
4531                        \__enumext_store_level_close_vii:
4532                    }
4533            }
4534    }
```

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

12.43.1   The command `\item` in enumext*

`\__enumext_first_item_tmp_vii:`
The `\__enumext_first_item_tmp_vii:` function will remove horizontal space equal to `\labelwidth` plus `\labelsep` to the left of the first `\item` in the environment at the point of execution of this function, where it is equal to the `\__enumext_stop_item_tmp_vii:` function inside the environment body definition.

```
4535  \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_vii:
4536    {
4537      \skip_horizontal:n { -\l__enumext_labelwidth_vii_dim - \l__enumext_labelsep_vii_dim }
4538    }
```

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

`\__enumext_start_item_tmp_vii:`
First we will call the function `\__enumext_stop_item_tmp_vii:` that we will redefine later, we will 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`.

```
4539  \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
4540    {
4541      \__enumext_stop_item_tmp_vii:
4542      \int_incr:N \l__enumext_item_column_pos_vii_int
4543      \int_gincr:N \g__enumext_item_count_all_vii_int
4544      \__enumext_item_peek_args_vii:
4545    }
```

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

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

```
4546  \cs_new_protected:Nn \__enumext_item_peek_args_vii:
4547    {
4548      \peek_meaning:NTF (
4549        { \__enumext_joined_item_vii:w }
4550        { \__enumext_joined_item_vii:w (1) }
4551    }
```

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

`\__enumext_joined_item_vii:w`
The function `\__enumext_joined_item_vii:w` will first call the function `\__enumext_starred_-joined_item_vii:n` in charge of setting the *width* of the box that will store the content passed to `\item`. Then we will look for the argument "`*`", if it is present we will call the function `\__enumext_starred_item_vii:w` otherwise we will call the function `\__enumext_standar_item_vii:w`.

```
4552  \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
4553    {
4554      \__enumext_starred_joined_item_vii:n {#1}
4555      \peek_meaning_remove:NTF *
4556        { \__enumext_starred_item_vii:w  }
4557        { \__enumext_standar_item_vii:w }
4558    }
```

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

`\__enumext_standar_item_vii:w`
The function `\__enumext_standar_item_vii:w` will first look for the argument "`[`", if present it will set the state of the variable `\l__enumext_wrap_label_opt_vii_bool` equal to the state of the variable `\l__enumext_wrap_label_opt_vii_bool` handled by the key `wrap-label*` and finally execute the *non-enumerated* version `\item[`⟨*custom*⟩`]` by means of the function `\__enumext_start_item_vii:w`, otherwise we will set the value of the variable `\l__enumext_wrap_label_vii_bool` handled by the `wrap-label` key to true and set the switch `\if@noitemarg` to true to execute the enumerated version of `\item` by means of the function `\__enumext_start_item_vii:w` [`\l__enumext_label_vii_tl`].

```
4559  \cs_new_protected:Npn \__enumext_standar_item_vii:w
4560    {
4561      \bool_set_false:N \l__enumext_item_starred_vii_bool
4562      \peek_meaning:NTF [
4563        {
4564          \bool_set_eq:NN \l__enumext_wrap_label_vii_bool \l__enumext_wrap_label_opt_vii_bool
4565          \__enumext_start_item_vii:w
4566        }
4567        {
4568          \bool_set_true:N \l__enumext_wrap_label_vii_bool
```

```
4569            \legacy_if_set_true:n { @noitemarg }
4570            \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4571          }
4572      }
```

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

`\__enumext_starred_item_vii:w`
`\__enumext_starred_item_vii_aux_i:w`
`\__enumext_starred_item_vii_aux_ii:w`
`\__enumext_starred_item_vii_aux_iii:w`

The function `\__enumext_starred_item_vii:w` together with the specified auxiliary functions `aux_i:w`, `aux_ii:w`, and `aux_iii:w` execute `\item*`, `\item*[`⟨*symbol*⟩`]` and `\item*[`⟨*symbol*⟩`][`⟨*offset*⟩`]`.

```
4573  \cs_new_protected:Npn \__enumext_starred_item_vii:w
4574    {
4575      \bool_set_true:N \l__enumext_item_starred_vii_bool
4576      \bool_set_true:N \l__enumext_wrap_label_vii_bool
4577      \peek_meaning:NTF [
4578        { \__enumext_starred_item_vii_aux_i:w }
4579        { \__enumext_starred_item_vii_aux_ii:w }
4580    }
4581  \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
4582    {
4583      \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
4584      \__enumext_starred_item_vii_aux_ii:w
4585    }
4586  \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
4587    {
4588      \peek_meaning:NTF [
4589        { \__enumext_starred_item_vii_aux_iii:w }
4590        {
4591          \dim_set_eq:NN \l__enumext_item_symbol_sep_vii_dim \l__enumext_labelsep_vii_dim
4592          \legacy_if_set_true:n { @noitemarg }
4593          \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4594        }
4595    }
4596  \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
4597    {
4598      \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
4599      \legacy_if_set_true:n { @noitemarg }
4600      \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ]
4601    }
```

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

`\__enumext_fake_make_label_vii:n`

The `\__enumext_fake_make_label_vii:n` function will be in charge of handling our definition of `\item`. First we increment the counter `enumXvii` for the enumerated items and activate support for the *check answers* mechanism, followed by support for `\item*[`⟨*symbol*⟩`][`⟨*offset*⟩`]` if present, then the `wrap-label` and `wrap-label*` keys which we execute using `\makebox` whose width will be given by the `labelwidth` key and position by the `align` key, inside the argument of this we will execute the `font` key together with the function defined by the `wrap-label` or `wrap-label*` keys. Finally we execute the `labelsep` key applying a *horizontal space*.

```
4602  \cs_new_protected_nopar:Npn \__enumext_fake_make_label_vii:n #1
4603    {
4604      \legacy_if:nT { @noitemarg }
4605        {
4606          \legacy_if_set_false:n { @noitemarg }
4607          \legacy_if:nT { @nmbrlist }
4608            {
4609              \refstepcounter{enumXvii}
4610              \bool_if:NT \l__enumext_check_answers_bool
4611                {
4612                  \int_gincr:N \g__enumext_item_number_int
4613                  \bool_set_true:N \l__enumext_item_number_bool
4614                }
4615            }
4616        }
4617      \bool_if:NT \l__enumext_item_starred_vii_bool
4618        {
4619          \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
4620            {
4621              \tl_gset_eq:NN
4622                \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
4623            }
```

```
4624          \mode_leave_vertical:
4625          \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
4626          \hbox_overlap_left:n { \g__enumext_item_symbol_aux_vii_tl }
4627          \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
4628          \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
4629        }
4630      \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
4631        {
4632          \tl_use:N \l__enumext_label_font_style_vii_tl
4633          \bool_if:NTF \l__enumext_wrap_label_vii_bool
4634            {
4635              \__enumext_wrapper_label_vii:n {#1}
4636            }
4637            { #1 }
4638        }
4639      \skip_horizontal:N \l__enumext_labelsep_vii_dim
4640    }
```

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

12.43.2   Real definition of \item in enumext*

The functions __enumext_start_item_vii:w and __enumext_stop_item_vii: executing the true definition of \item inside the enumext* environment, unlike the implementation in shortlst we will NOT use an extra group and the plain form of the lrbox environment.

__enumext_start_item_vii:w   The first thing we will do is set the value of __enumext_stop_item_tmp_vii: equal to __enumext_-stop_item_vii: which we will define later, after that we will start capturing \item and its ⟨contents⟩ in a *horizontal box* where the width will be \itemwidth plus \labelwidth plus \labelsep.

```
4641  \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
4642    {
4643      \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
4644      \hbox_set_to_wd:Nnw \l__enumext_item_text_vii_box
4645        {
4646          \l__enumext_joined_width_vii_dim
4647          + \l__enumext_labelwidth_vii_dim
4648          + \l__enumext_labelsep_vii_dim
4649        }
```

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

```
4650        \IfDocumentMetadataTF { }
4651          {
4652            \bool_if:NF \l__enumext_footnotes_key_bool
4653              {
4654                \__enumext_renew_footnote:
4655              }
4656          }
```

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

```
4657        \__enumext_start_list_tag:n {enumext*}
4658        \__enumext_fake_make_label_vii:n {#1}
4659        \__enumext_stop_start_list_tag:
```

Finally we open the minipage environment capture the ⟨item content⟩ and execute first and itemindent keys, then listparindent key which will be equal to \parindent, then parsep key which will be equal to \parskip.

```
4660        \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim  }
4661          \tl_use:N \l__enumext_after_list_args_vii_tl
4662          \tl_use:N \l__enumext_fake_item_indent_vii_tl
4663          \dim_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
4664          \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
4665    }
```

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

__enumext_stop_item_vii: The __enumext_stop_item_vii: function will finish the fetching \item and its ⟨content⟩ by closing the minipage environment, the *sockets* for *tagging* PDF and the *horizontal box*.

```
4666  \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
4667    {
4668      \__enumext_endminipage:
4669      \__enumext_stop_list_tag:n {enumext*}
4670      \hbox_set_end:
```

Here we will reduce the *warnings* a bit by setting the value of \hbadness to 10000, print the ⟨*contents*⟩ of the *box* along with \footnote.

```
4671    \int_set:Nn \hbadness { 10000 }
4672    \box_use_drop:N \l__enumext_item_text_vii_box
4673    \IfDocumentMetadataTF { }
4674      {
4675        \bool_if:NF \l__enumext_footnotes_key_bool
4676          {
4677            \__enumext_print_footnote:
4678          }
4679      }
```

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

```
4680    \int_compare:nNnTF
4681      { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
4682      {
4683        \par\noindent
4684        \int_zero:N \l__enumext_item_column_pos_vii_int
4685      }
4686      {
4687        \skip_horizontal:N \l__enumext_columns_sep_vii_dim
4688      }
4689    }
```

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

__enumext_remove_extra_parsep_vii: Remove the *vertical space* equal to \parsep=\itemsep when the total number of items is divisible by the number of items in the last row of the environment. Here the use of \unskip or \removelastskip fails and does not obtain the expected result, using \vspace is the option and in this case, we can use a simplified version since we are always in ⟨*vertical mode*⟩.

```
4690    \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
4691      {
4692        \int_compare:nNnT
4693          {
4694            \int_mod:nn
4695              {  \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
4696          }
4697        =
4698        { 0 }
4699        {
4700          \para_end:
4701          \skip_vertical:n { -\l__enumext_itemsep_vii_skip }
4702          \skip_vertical:N \c_zero_skip
4703          \int_gzero:N \g__enumext_item_count_all_vii_int
4704        }
4705      }
```

(*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.

```
4706    \__enumext_after_env:nn {enumext*}
4707      {
4708        \__enumext_execute_after_env:
4709      }
```

12.44   The environment keyans*

keyans*   First we will generate the environment and we will give a temporary definition to __enumext_stop_item_-tmp_viii: equal to __enumext_first_item_tmp_viii: and next to \item equal to __enumext_-start_item_tmp_viii: which we will redefine later. The implementation of this environment is the same as that used by the enumext* environment except for the __enumext_check_starred_cmd:n function added in the second part.

```
4710    \NewDocumentEnvironment{keyans*}{ o }
4711      {
4712        \__enumext_safe_exec_viii:
4713        \__enumext_parse_keys_viii:n {#1}
4714        \__enumext_before_list_viii:
4715        \__enumext_start_list:nn { }
4716          {
4717            \__enumext_list_arg_two_viii:
```

```
4718        \__enumext_before_keys_exec_viii:
4719      }
4720    % Stop tagging
4721    \IfDocumentMetadataTF { \tag_suspend:n {keyans*} } { }
4722    \__enumext_starred_columns_set_viii:
4723    \item[] \scan_stop:
4724    \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_first_item_tmp_viii:
4725    \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
4726    \ignorespaces
4727  }
4728  {
4729    \IfDocumentMetadataTF { \tag_struct_end:n {tag=text-unit} } { }
4730    \__enumext_stop_item_tmp_viii:
4731    \__enumext_remove_extra_parsep_viii:
4732    \__enumext_check_starred_cmd:n { item }
4733    \__enumext_after_list_viii:
4734  }
```

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

__enumext_safe_exec_viii: The __enumext_safe_exec_viii: function will first check if the save-ans key is active and only when this is true the environment will be available, it will increment the value of \l__enumext_keyans_level_h_int and return an error message when we are nesting the environment, then it will call the __enumext_keyans_name_and_start: function in charge of saving the name of the environment and the line it is running on, then it will check if we are trying to nest keyans* in enumext* returning an error and we will set \l__enumext_starred_bool to true, finally we will check if we are within the appropriate level within the enumext environment.

```
4735  \cs_new_protected:Nn \__enumext_safe_exec_viii:
4736    {
4737      \bool_if:NF \l__enumext_store_active_bool
4738        {
4739          \msg_error:nnnn { enumext } { wrong-place }{ keyans* }{ save-ans }
4740        }
4741      \int_incr:N \l__enumext_keyans_level_h_int
4742      \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
4743        {
4744          \msg_error:nn { enumext } { nested }
4745        }
4746      \__enumext_keyans_name_and_start:
4747      \bool_if:NT \l__enumext_starred_bool
4748        {
4749          \msg_error:nnn { enumext } { nested-horizontal } { enumext* }
4750        }
4751      \bool_set_true:N \l__enumext_starred_bool
4752      % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
4753      \bool_set_false:N \l__enumext_store_active_bool
4754      \int_compare:nNnT { \l__enumext_level_int } > { 1 }
4755        {
4756          \msg_error:nn { enumext } { keyans-wrong-level }
4757        }
4758    }
```

(*End of definition for* __enumext_safe_exec_viii:.)

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

```
4759  \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
4760    {
4761      \tl_if_novalue:nF {#1}
4762        {
4763          \keys_set:nn { enumext / keyans* } {#1}
4764        }
4765    }
```

(*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.

```
4766  \cs_new_protected:Nn \__enumext_before_list_viii:
4767    {
```

```
4768        \__enumext_vspace_above_viii:
4769        \__enumext_before_args_exec_viii:
4770        \__enumext_start_mini_viii:
4771    }
```

(*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.

```
4772 \cs_new_protected:Nn \__enumext_after_list_viii:
4773    {
4774        \__enumext_stop_mini_viii:
4775        \__enumext_after_stop_list_viii:
4776        \__enumext_vspace_below_viii:
4777    }
```

(*End of definition for* __enumext_after_list_viii:.)

12.44.1 The command \item in keyans*

The idea here is to make the \item command behave in the same way as in the keyans environment with the difference of the *optional argument* (⟨*number*⟩) which works in the same way as in the enumext* environment. In simple terms we want to store the ⟨*label*⟩ next to the [⟨*content*⟩] if it is present in the ⟨*sequence*⟩ and ⟨*prop list*⟩ defined by save-ans key for \item*, \item*[⟨*content*⟩], \item(⟨*number*⟩)* and \item(⟨*number*⟩)*[⟨*content*⟩] commands.

__enumext_first_item_tmp_viii: The __enumext_first_item_tmp_viii: function will remove horizontal space equal to \labelwidth plus \labelsep to the left of the first \item in the environment at the point of execution of this function, where it is equal to the __enumext_stop_item_tmp_viii: function inside the environment body definition.

```
4778 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_viii:
4779    {
4780        \skip_horizontal:n { -\l__enumext_labelwidth_viii_dim - \l__enumext_labelsep_viii_dim }
4781    }
```

(*End of definition for* __enumext_first_item_tmp_viii:.)

__enumext_start_item_tmp_viii: First we will call the function __enumext_stop_item_tmp_viii: that we will redefine later, we will increment the value of \l__enumext_item_column_pos_viii_int that will count the item's by rows and the value of \g__enumext_item_count_all_viii_int that will count the total of item's in the environment. After that we will call the function __enumext_item_peek_args_viii: that will handle the arguments passed to \item.

```
4782 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
4783    {
4784        \__enumext_stop_item_tmp_viii:
4785        \int_incr:N \l__enumext_item_column_pos_viii_int
4786        \int_gincr:N \g__enumext_item_count_all_viii_int
4787        \__enumext_item_peek_args_viii:
4788    }
```

(*End of definition for* __enumext_start_item_tmp_viii:.)

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

```
4789 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
4790    {
4791        \peek_meaning:NTF (
4792            { \__enumext_joined_item_viii:w }
4793            { \__enumext_joined_item_viii:w (1) }
4794    }
```

(*End of definition for* __enumext_item_peek_args_viii:.)

__enumext_joined_item_viii:w   The function __enumext_joined_item_viii:w will first call the function __enumext_starred_-joined_item_viii:n in charge of setting the *width* of the box that will store the content passed to \item. Then we will look for the argument "*", if it is present we will call the function __enumext_starred_-item_viii:w otherwise we will call the function __enumext_standar_item_viii:w.

```
4795 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
4796    {
4797        \__enumext_starred_joined_item_viii:n {#1}
```

```
4798      \peek_meaning_remove:NTF *
4799        { \__enumext_starred_item_viii:w  }
4800        { \__enumext_standar_item_viii:w }
4801      }
```

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

__enumext_standar_item_viii:w   The function __enumext_standar_item_viii:w will first look for the argument "[", if present it will set the state of the variable \l__enumext_wrap_label_opt_viii_bool equal to the state of the variable \l__enumext_wrap_label_opt_viii_bool handled by the key wrap-label* and finally execute the *non-enumerated* version \item[⟨custom⟩] by means of the function __enumext_start_item_viii:w, otherwise we will set the value of the variable \l__enumext_wrap_label_viii_bool handled by the wrap-label key to true and set the switch \if@noitemarg to true to execute the enumerated version of \item by means of the function __enumext_start_item_viii:w [\l__enumext_label_viii_tl].

```
4802  \cs_new_protected:Npn \__enumext_standar_item_viii:w
4803    {
4804      \bool_set_false:N \l__enumext_item_starred_viii_bool
4805      \peek_meaning:NTF [
4806        {
4807          \bool_set_eq:NN \l__enumext_wrap_label_viii_bool \l__enumext_wrap_label_opt_viii_bool
4808          \__enumext_start_item_viii:w
4809        }
4810        {
4811          \bool_set_true:N \l__enumext_wrap_label_viii_bool
4812          \legacy_if_set_true:n { @noitemarg }
4813          \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4814        }
4815      }
```

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

__enumext_starred_item_viii:w   The function __enumext_starred_item_viii:w together with the specified auxiliary functions aux_i:w
__enumext_starred_item_viii_aux_i:w   and aux_ii:w execute \item* and \item*[⟨content⟩].
__enumext_starred_item_viii_aux_ii:w

```
4816  \cs_new_protected:Npn \__enumext_starred_item_viii:w
4817    {
4818      \bool_set_true:N \l__enumext_item_starred_viii_bool
4819      \bool_set_true:N \l__enumext_wrap_label_viii_bool
4820      \peek_meaning:NTF [
4821        { \__enumext_starred_item_viii_aux_i:w }
4822        { \__enumext_starred_item_viii_aux_ii:w }
4823      }
```

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.

```
4824  \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
4825    {
4826      \tl_clear:N \l__enumext_store_current_label_tl
4827      \tl_if_novalue:nF { #1 }
4828        {
4829          \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
4830            {
4831              \tl_put_right:Ne \l__enumext_store_current_label_tl
4832                {
4833                  \l__enumext_store_keyans_item_opt_sep_tl
4834                }
4835              \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
4836            }
4837          \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
4838        }
4839      \__enumext_starred_item_viii_aux_ii:w
4840    }
4841  \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
4842    {
4843      \legacy_if_set_true:n { @noitemarg }
4844      \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ]
4845    }
```

(*End of definition for* __enumext_starred_item_viii:w, __enumext_starred_item_viii_aux_i:w, *and* __enumext_starred_item_viii_aux_ii:w.)

__enumext_starred_item_exec:

The function __enumext_starred_item_exec: will be in charge of storing the current ⟨label⟩ for \item* followed by the [⟨content⟩] for \item*[⟨content⟩] if present in the ⟨sequence⟩ and ⟨prop list⟩ set by the save-ans key. In this same function the keys show-ans, show-pos and save-ref are implemented.

```
4846 \cs_new_protected:Nn \__enumext_starred_item_exec:
4847   {
4848     \tl_put_left:Ne \l__enumext_store_current_label_tl { \l__enumext_label_viii_tl }
4849     \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
4850     \__enumext_keyans_store_ref:
4851     \tl_put_left:Ne \l__enumext_store_current_label_tl { \item }
4852     \__enumext_keyans_addto_seq_link:
4853     \int_gincr:N \g__enumext_check_starred_cmd_int
4854     \bool_if:NT \l__enumext_show_answer_bool
4855       {
4856         \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4857       }
4858     \bool_if:NT \l__enumext_show_position_bool
4859       {
4860         \tl_set:Ne \l__enumext_mark_answer_sym_tl
4861           {
4862             \group_begin:
4863               \exp_not:N \normalfont
4864               \exp_not:N \footnotesize [ \int_eval:n
4865                 {
4866                   \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
4867                 }
4868               ]
4869             \group_end:
4870           }
4871         \__enumext_print_keyans_box:NN \l__enumext_labelwidth_i_dim \l__enumext_labelsep_i_dim
4872       }
4873   }
```

(*End of definition for* __enumext_starred_item_exec:.)

__enumext_fake_make_label_viii:n

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

```
4874 \cs_new_protected_nopar:Npn \__enumext_fake_make_label_viii:n #1
4875   {
4876     \legacy_if:nT { @noitemarg }
4877       {
4878         \legacy_if_set_false:n { @noitemarg }
4879         \legacy_if:nT { @nmbrlist }
4880           {
4881             \refstepcounter{enumXviii}
4882           }
4883       }
4884     \bool_if:NT \l__enumext_item_starred_viii_bool
4885       {
4886         \__enumext_starred_item_exec:
4887       }
4888     \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
4889       {
4890         \tl_use:N \l__enumext_label_font_style_viii_tl
4891         \bool_if:NTF \l__enumext_wrap_label_viii_bool
4892           {
4893             \__enumext_wrapper_label_viii:n {#1}
4894           }
4895           { #1 }
4896       }
4897     \skip_horizontal:N \l__enumext_labelsep_viii_dim
4898   }
```

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

12.44.2 Real definition of \item in keyans*

__enumext_start_item_viii:w

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

```
4899 \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
4900   {
4901     \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
4902     \hbox_set_to_wd:Nnw \l__enumext_item_text_viii_box
4903       {
4904         \l__enumext_joined_width_viii_dim
```

```
4905        + \l__enumext_labelwidth_viii_dim
4906        + \l__enumext_labelsep_viii_dim
4907      }
4908      \IfDocumentMetadataTF { }
4909        {
4910          \bool_if:NF \l__enumext_footnotes_key_bool
4911            {
4912              \__enumext_renew_footnote:
4913            }
4914        }
4915      \__enumext_start_list_tag:n {keyans*}
4916      \__enumext_fake_make_label_viii:n {#1}
4917      \__enumext_stop_start_list_tag:
4918      \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim  }
4919        \tl_use:N \l__enumext_after_list_args_viii_tl
4920        \bool_if:NT \l__enumext_item_starred_viii_bool
4921          {
4922            \tl_use:N \l__enumext_fake_item_indent_viii_tl
4923            \__enumext_keyans_show_item_opt:
4924            \skip_horizontal:n { -\l__enumext_fake_item_indent_viii_dim - \l__enumext_labelsep_vi
4925          }
4926          {
4927            \tl_use:N \l__enumext_fake_item_indent_viii_tl
4928          }
4929        \dim_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
4930        \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
4931      }
```

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

__enumext_stop_item_viii: The __enumext_stop_item_viii: function will finish the fetching \item and its ⟨*content*⟩ by closing the minipage environment and the *horizontal box*. Here we will reduce the *warnings* a bit by setting the value of \hbadness to 10000, print the ⟨*contents*⟩ of the *box* along with \footnote and finally set the vertical and horizontal spaces between rows and columns.

```
4932  \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
4933    {
4934        \__enumext_endminipage:
4935      \__enumext_stop_list_tag:n {keyans*}
4936      \hbox_set_end:
4937      \int_set:Nn \hbadness { 10000 }
4938      \box_use_drop:N \l__enumext_item_text_viii_box
4939      \IfDocumentMetadataTF { }
4940        {
4941          \bool_if:NF \l__enumext_footnotes_key_bool
4942            {
4943              \__enumext_print_footnote:
4944            }
4945        }
4946      \int_compare:nNnTF
4947        { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
4948        {
4949          \par\noindent
4950          \int_zero:N \l__enumext_item_column_pos_viii_int
4951        }
4952        {
4953          \skip_horizontal:N \l__enumext_columns_sep_viii_dim
4954        }
4955    }
```

(*End of definition for __enumext_stop_item_viii:.*)

__enumext_remove_extra_parsep_viii: Finally we will remove the *vertical space* equal to \parsep when the total number of items is divisible by the number of items in the last row of the environment.

```
4956  \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
4957    {
4958      \int_compare:nNnT
4959        {
4960          \int_mod:nn
4961            { \g__enumext_item_count_all_viii_int }
4962            { \l__enumext_columns_viii_int }
4963        }
```

```
4964        =
4965          { 0 }
4966          {
4967            \para_end:
4968            \skip_vertical:n { -\l__enumext_itemsep_viii_skip }
4969            \skip_vertical:N \c_zero_skip
4970            \int_gzero:N \g__enumext_item_count_all_viii_int
4971          }
4972      }
```

(*End of definition for* __enumext_remove_extra_parsep_viii:.)

12.45   The command \getkeyans

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

```
4973 \NewDocumentCommand \getkeyans { m }
4974   {
4975     \exp_args:Ne \__enumext_getkeyans_aux:n
4976       { \tl_to_str:e { \text_expand:n {#1} } }
4977   }
```

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

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

```
4978 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
4979   {
4980     \str_if_in:nnTF {#1} { : }
4981       {
4982         \use:e
4983           {
4984             \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
4985               { {##1} {##2} }
4986           }
4987         \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
4988       }
4989       { \msg_error:nnn { enumext } { missing-colon } {#1} }
4990   }
```

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

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

```
4991 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
4992   {
4993     \prop_if_exist:cTF { g__enumext_#1_prop }
4994       {
4995         \prop_item:cn { g__enumext_#1_prop }{#2}
4996       }
4997       {
4998         \msg_error:nnn { enumext } { undefined-storage-anskey } {#1}
4999       }
5000   }
```

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

12.46   The command \printkeyans

The \printkeyans command prints "*all stored content*" in the ⟨*sequence*⟩ defined by the save-ans key.
The first thing we will do is define a set of ⟨*filtered keys*⟩ with which we will control the options of the
different nesting levels for the environment enumext and enumext* by storing their values in the list of tokens
\l__enumext_print_keyans_X_tl.

The variable \l__enumext_print_keyans_starred_tl will have the default ⟨*keys*⟩ for \printkeyans*
and will be set by \setenumext[⟨*print**⟩] and the variable \l__enumext_print_keyans_vii_tl will
have the default keys for the environment enumext* nested within the ⟨*sequence*⟩ and will be set by
\setenumext[⟨*print ,**⟩], the rest of the variables will be for the environment enumext and will be set
by \setenumext[⟨*print , level*⟩].

```
5001 \keys_define:nn  { enumext / print }
5002   {
```

```
5003        print*  .code:n      = \keys_precompile:neN { enumext / enumext* }
5004                                { \__enumext_filter_save_key:n {#1} }
5005                                \l__enumext_print_keyans_starred_tl, % starred cmd
5006        print*  .initial:n  = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
5007        print-1 .code:n      = \keys_precompile:neN { enumext / level-1 }
5008                                { \__enumext_filter_save_key:n {#1} }
5009                                \l__enumext_print_keyans_i_tl,
5010        print-1 .initial:n  = { nosep, label=\arabic*., columns=2, first=\small, font=\small },
5011        print-2 .code:n      = \keys_precompile:neN { enumext / level-2 }
5012                                { \__enumext_filter_save_key:n {#1} }
5013                                \l__enumext_print_keyans_ii_tl,
5014        print-2 .initial:n  = { nosep, label=(\alph*), first=\small, font=\small },
5015        print-3 .code:n      = \keys_precompile:neN { enumext / level-3 }
5016                                { \__enumext_filter_save_key:n {#1} }
5017                                \l__enumext_print_keyans_iii_tl,
5018        print-3 .initial:n  = { nosep, label=\roman*., first=\small, font=\small },
5019        print-4 .code:n      = \keys_precompile:neN { enumext / level-4 }
5020                                { \__enumext_filter_save_key:n {#1} }
5021                                \l__enumext_print_keyans_iv_tl,
5022        print-4 .initial:n  = { nosep, label=\Alph*., first=\small, font=\small },
5023        print-* .code:n      = \keys_precompile:neN { enumext / enumext* }
5024                                { \__enumext_filter_save_key:n {#1} }
5025                                \l__enumext_print_keyans_vii_tl, % starred nested
5026        print-* .initial:n  = { nosep, label=\arabic*., first=\small, font=\small },
5027      }
```

💣 The reason for storing ⟨keys⟩ in token lists using \keys_precompile:neN is because the keys are set via \setenumext but are later executed by running the command \printkeyans and they are not handled directly by its *optional argument*, except those related to the *first* opening level.

\printkeyans   Create a user command to print *"all stored content"* in ⟨sequence⟩ for \anskey, anskey*, \item* and \anspic*. Within a group we will run our *"precompiled keys"* and then call the internal function __enumext_printkeyans:nnn.

```
5028  \NewDocumentCommand \printkeyans { s O{} m }
5029    {
5030      \group_begin:
5031        \tl_use:N \l__enumext_print_keyans_i_tl
5032        \tl_use:N \l__enumext_print_keyans_ii_tl
5033        \tl_use:N \l__enumext_print_keyans_iii_tl
5034        \tl_use:N \l__enumext_print_keyans_iv_tl
5035        \tl_use:N \l__enumext_print_keyans_vii_tl
5036        \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
5037      \group_end:
5038    }
```

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

__enumext_printkeyans:nnn   The internal function __enumext_printkeyans:nnn will check for the existence of the ⟨sequence⟩, if it does not exist it will return an error message, then it will check if not empty.

```
5039  \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
5040    {
5041      \seq_if_exist:cTF { g__enumext_#3_seq }
5042        {
5043          \seq_if_empty:cF { g__enumext_#3_seq }
5044            {
```

If the *starred argument* '*' is present we will check that the environment enumext* is not saved in the ⟨sequence⟩, then execute the variable \l__enumext_print_keyans_starred_tl that contains the default ⟨keys⟩ for the environment enumext*, it will open the environment enumext* passing the *optional argument* to the *"first level"*, set the key base-fix and then will map the ⟨sequence⟩.

```
5045              \bool_if:nTF {#1}
5046                {
5047                  \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
5048                    {
5049                      \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
5050                    }
5051                    {
5052                      \tl_use:N \l__enumext_print_keyans_starred_tl
5053                      \begin{enumext*}[#2]
5054                        \keys_set:nn { enumext / level-1 }{ base-fix }
5055                        \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
```

```
5056                        \end{enumext*}
5057                      }
5058                 }
```

Otherwise it will open the environment enumext passing the *optional argument* to the *"first level"*, set the key base-fix and then map the ⟨*sequence*⟩.

```
5059                 {
5060                   \begin{enumext}[#2]
5061                     \keys_set:nn { enumext / enumext* }{ base-fix }
5062                     \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5063                   \end{enumext}
5064                 }
5065             }
5066         }
5067         {
5068           \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
5069         }
5070     }
```

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

12.47 The command \setenumext

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

__enumext_filter_first_level:n
__enumext_filter_first_level_key:n
__enumext_filter_first_level_pair:nn

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

```
5071 \cs_new:Npn \__enumext_filter_first_level:n #1
5072   {
5073     \use:e
5074       {
5075         \keyval_parse:NNn
5076           \__enumext_filter_first_level_key:n
5077           \__enumext_filter_first_level_pair:nn {#1}
5078       }
5079   }
```

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*.

```
5080 \cs_new:Npn \__enumext_filter_first_level_key:n #1
5081   {
5082     \str_case:nnF {#1}
5083       {
5084         { resume   } {}
5085         { resume*  } {}
5086       }
5087       { , { \exp_not:n {#1} } }
5088   }
```

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.

```
5089 \cs_new:Npn \__enumext_filter_first_level_pair:nn #1#2
5090   {
5091     \str_case:nnF {#1}
5092       {
5093         { series } {}
5094         { resume } {}
5095         { save-ans } {}
5096       }
5097       { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
5098   }
```

(*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.

```
5099 \keys_define:nn { enumext / meta-families }
5100   {
5101     enumext-1 .code:n =
5102             {
5103               \keys_set:ne { enumext / level-1 }
```

```
5104                          {
5105                              \__enumext_filter_first_level:n {#1}
5106                          }
5107                      } ,
5108          enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
5109          enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
5110          enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
5111          keyans    .code:n = { \keys_set:nn { enumext / keyans  } {#1} } ,
5112          enumext*  .code:n =
5113                      {
5114                          \keys_set:ne { enumext / enumext* }
5115                              {
5116                                  \__enumext_filter_first_level:n {#1}
5117                              }
5118                      } ,
5119          keyans*   .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
5120          print*    .code:n = { \keys_set:nn { enumext / print   } { print*  = {#1} } } ,
5121          print-1   .code:n = { \keys_set:nn { enumext / print   } { print-1 = {#1} } } ,
5122          print-2   .code:n = { \keys_set:nn { enumext / print   } { print-2 = {#1} } } ,
5123          print-3   .code:n = { \keys_set:nn { enumext / print   } { print-3 = {#1} } } ,
5124          print-4   .code:n = { \keys_set:nn { enumext / print   } { print-4 = {#1} } } ,
5125          print-*   .code:n = { \keys_set:nn { enumext / print   } { print-* = {#1} } } ,
5126          unknown   .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
5127      }
```

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

```
5128  \seq_const_from_clist:Nn \c__enumext_all_families_seq
5129      {
5130          enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
5131          keyans*, print-1, print-2, print-3, print-4, print-*, print*,
5132      }
```

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

```
5133  \NewDocumentCommand \setenumext { O{enumext,1} +m }
5134      {
5135          \seq_clear:N \l__enumext_setkey_tmpa_seq
5136          \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
5137          \int_set:Nn \l__enumext_setkey_tmpa_int
5138              {
5139                  \seq_count:N \l__enumext_setkey_tmpb_seq
5140              }
5141          \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
5142              {
5143                  \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
5144                  \seq_map_function:NN \l__enumext_setkey_tmpb_seq \__enumext_set_parse:n
5145                  \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
5146                      {
5147                          \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
5148                      }
5149              }
5150              {
5151                  \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
5152              }
5153          \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
5154              { \seq_map_inline:Nn \c__enumext_all_families_seq }
5155              { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
5156              {
5157                  \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
5158              }
5159      }
```

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

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

```
5160  \cs_new_protected:Npn \__enumext_set_parse:n #1
5161      {
5162          \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
5163          \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
5164              { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
5165          \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
5166              {
```

```
5167        \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
5168          { \tl_trim_spaces:n {#1} }
5169        }
5170      { \__enumext_set_error:nn {#1} { } }
5171    }
5172 \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
5173    { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }
```

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

12.48 The command \setenumextmeta

The command \setenumextmeta will be responsible for adding new *"meta-keys"* for the enumext and enumext* environments. The implementation code was given by Jonathan P. Spratte (@Skillmon) answer in Add .meta key to existing keys (l3keys).

\setenumextmeta
\c__enumext_meta_paths_prop
__enumext_add_meta_key:nnn
__enumext_def_meta_key:nnn
__enumext_def_meta_key:Vnn

First we will create a prop list \c__enumext_meta_paths_prop to handle the *optional argument*.

```
5174 \prop_const_from_keyval:Nn \c__enumext_meta_paths_prop
5175    {
5176      {enumext,1} = level-1,
5177      {enumext,2} = level-2,
5178      {enumext,3} = level-3,
5179      {enumext,4} = level-4,
5180      {enumext*}  = enumext*
5181    }
```

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

```
5182 \NewDocumentCommand \setenumextmeta { s O{enumext,1} m +m }
5183    {
5184      \str_if_eq:eeTF { \tl_trim_spaces:n {#3} } { unknown }
5185        { \msg_error:nn { enumext } { prohibited-unknown } }
5186        {
5187          \bool_if:nTF {#1}
5188            {
5189              \int_step_inline:nn { 4 }
5190                { \__enumext_add_meta_key:nnn { enumext, ##1 } {#3} {#4} }
5191              \__enumext_add_meta_key:nnn { enumext* } {#3} {#4}
5192            }
5193            { \__enumext_add_meta_key:nnn {#2} {#3} {#4} }
5194        }
5195    }
```

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

```
5196 \cs_new_protected:Npn \__enumext_add_meta_key:nnn #1
5197    {
5198      \tl_set:Nn \l__enumext_meta_path_tl {#1}
5199      \tl_replace_all:Nnn \l__enumext_meta_path_tl { ~ } {}
5200      \prop_get:NVNTF
5201        \c__enumext_meta_paths_prop \l__enumext_meta_path_tl \l__enumext_meta_path_tl
5202        { \__enumext_def_meta_key:Vnn \l__enumext_meta_path_tl }
5203        {
5204          \msg_error:nnn { enumext } { unknown-set } {#1}
5205          \use_none:nn
5206        }
5207    }
5208 \cs_new_protected:Npn \__enumext_def_meta_key:nnn #1#2#3
5209    {
5210      \bool_lazy_or:nnTF
5211        { \keys_if_exist_p:nn { enumext / #1 } {#2} }
5212        { \keys_if_exist_p:nn { enumext / enumext* } {#2} }
5213        { \msg_error:nnn { enumext } { already-defined } {#2} }
5214        {
5215          \keys_define:nn { enumext / #1 }
5216            {
5217              #2 .meta:n = {#3},
5218              #2 .value_forbidden:n = true
5219            }
5220        }
5221    }
5222 \cs_generate_variant:Nn \__enumext_def_meta_key:nnn { V }
```

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

12.49 The command \foreachkeyans

The command \foreachkeyans will execute a *loop* over the ⟨*prop list*⟩ and return its contents. The implementation code is adapted from the answer provided by Enrico Gregorio (@egreg) in Expand a .cs defined by key inside the function.

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.

```
5223 \keys_define:nn { enumext / foreach }
5224   {
5225     before  .tl_set:N  = \l__enumext_foreach_before_tl,
5226     before  .value_required:n = true,
5227     after   .tl_set:N  = \l__enumext_foreach_after_tl,
5228     after   .value_required:n = true,
5229     start   .int_set:N = \l__enumext_foreach_start_int,
5230     start   .value_required:n = true,
5231     stop    .int_set:N = \l__enumext_foreach_stop_int,
5232     stop    .value_required:n = true,
5233     step    .int_set:N = \l__enumext_foreach_step_int,
5234     step    .value_required:n = true,
5235     wrapper .cs_set_protected:Np = \__enumext_foreach_wrapper:n #1,
5236     wrapper .value_required:n = true,
5237     sep     .tl_set:N  = \l__enumext_foreach_sep_tl,
5238     sep     .value_required:n = true,
5239     unknown .code:n    = { \__enumext_parse_foreach_keys:n {#1} }
5240   }
5241 \keys_precompile:nnN { enumext / foreach }
5242   {
5243     before={},after={},start=1,step=1,stop=0,wrapper=#1,sep=
5244   }
5245   \g__enumext_foreach_default_keys_tl
```

Functions for handling unknown ⟨*keys*⟩.

```
5246 \cs_new_protected:Npn \__enumext_parse_foreach_keys:nn #1#2
5247   {
5248     \tl_if_blank:nTF {#2}
5249       {
5250         \msg_error:nnn { enumext } { for-key-unknown } {#1}
5251       }
5252       {
5253         \msg_error:nnnn { enumext } { for-key-value-unknown } {#1} {#2}
5254       }
5255   }
5256 \cs_new_protected:Npn \__enumext_parse_foreach_keys:n #1
5257   {
5258     \exp_args:NV \__enumext_parse_foreach_keys:nn \l_keys_key_str {#1}
5259   }
```

We create the command.

```
5260 \NewDocumentCommand \foreachkeyans { +O{} m }
5261   {
5262     \__enumext_foreach_keyans:nn {#1} {#2}
5263   }
```

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

```
5264 \cs_new_protected:Npn \__enumext_foreach_keyans:nn #1 #2
5265   {
5266     \tl_use:N \g__enumext_foreach_default_keys_tl
5267     \keys_set:nn { enumext / foreach } {#1}
5268     \tl_set:Nn \l__enumext_foreach_name_prop_tl {#2}
5269     \prop_if_exist:cF { g__enumext_#2_prop }
5270       {
5271         \msg_error:nnn { enumext } { undefined-storage-anskey } {#2}
5272       }
5273     \int_compare:nNnT { \l__enumext_foreach_stop_int } = { 0 }
5274       {
5275         \int_set:Nn \l__enumext_foreach_stop_int
5276           { \prop_count:c { g__enumext_#2_prop } }
5277       }
5278     \seq_clear:N \l__enumext_foreach_print_seq
5279     \int_step_function:nnnN
```

```
5280        { \l__enumext_foreach_start_int }
5281        { \l__enumext_foreach_step_int }
5282        { \l__enumext_foreach_stop_int }
5283      \__enumext_foreach_add_body:n
5284      \seq_use:NV \l__enumext_foreach_print_seq \l__enumext_foreach_sep_tl
5285    }
5286 \cs_new_protected:Npn \__enumext_foreach_add_body:n #1
5287    {
5288      \seq_put_right:Ne \l__enumext_foreach_print_seq
5289        {
5290          \exp_not:V \l__enumext_foreach_before_tl
5291          \__enumext_foreach_wrapper:n
5292            {
5293              \prop_item:cn { g__enumext_ \l__enumext_foreach_name_prop_tl _prop }{#1}
5294            }
5295          \exp_not:V \l__enumext_foreach_after_tl
5296        }
5297    }
```

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

12.50   Messages

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

```
5298 \msg_new:nnn { enumext } { package-load }
5299    {
5300      The ~ '#1' ~ package ~ is ~ already ~ loaded.
5301    }
5302 \msg_new:nnn { enumext } { package-not-load }
5303    {
5304      The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
5305    }
5306 \msg_new:nnn { enumext } { package-load-foot }
5307    {
5308      The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
5309    }
```

Message used in the creation of counters by enumext package.

```
5310 \msg_new:nnn { enumext } { counters }
5311    {
5312      The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
5313      package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
5314    }
```

Message used by align and mark-pos keys.

```
5315 \msg_new:nnn { enumext } { unknown-choice }
5316    {
5317      The ~ value ~ '#3' ~ for ~ '#1' ~ key ~ is ~ invalid ~ use ~ ('#2').
5318    }
```

Message used by reserved anskey* environment by enumext package.

```
5319 \msg_new:nnnn { enumext } { anskey-env-error }
5320    {
5321      The ~ '#1' ~ environment ~is ~ reserved ~ by ~\\
5322      'enumext' ~ package, ~ It~ is~ already~ defined.
5323    }
5324    {
5325      The ~ anskey* ~ environment ~ is ~ defined ~ internally ~
5326      for ~ the ~ 'save-ans' ~ key.\\
5327    }
```

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

```
5328 \msg_new:nnn { enumext } { store-prop }
5329    {
5330      * ~ Package ~ enumext: ~ Creating ~
5331      \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
5332    }
5333 \msg_new:nnn { enumext } { store-seq }
5334    {
5335      * ~ Package ~ enumext: ~ Creating ~
5336      \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
5337    }
5338 \msg_new:nnn { enumext } { store-int }
```

```
5339      {
5340        * ~ Package ~ enumext: ~ Creating ~
5341        \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_context:.
5342      }
5343  \msg_new:nnn { enumext } { prop-seq-int-hook }
5344      {
5345        * ~ Package ~ enumext: ~ Elements ~ in ~
5346        \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
5347        * ~ Package ~ enumext: ~ Elements ~ in ~
5348        \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
5349        * ~ Package ~ enumext: ~ Value ~ off ~
5350        \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
5351      }
5352  \msg_new:nnn { enumext } { item-answer-hook }
5353      {
5354        * ~ Package ~ enumext: ~ Value ~ off ~
5355        \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\\
5356        * ~ Package ~ enumext: ~ Value ~ off ~
5357        \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\\
5358        * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
5359      }
```

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

```
5360  \msg_new:nnn { enumext } { invalid-key }
5361      {
5362        The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
5363      }
5364  \msg_new:nnn { enumext } { unknown-key-family }
5365      {
5366        Unknown~key~family~`\l_keys_key_str'~for~enumext.
5367      }
```

Messages used in length calculation.

```
5368  \msg_new:nnn { enumext } { width-negative }
5369      {
5370        Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\\
5371        The ~ key ~ '#1'~ accepts ~ values  ~ >= ~ 0pt.
5372      }
5373  \msg_new:nnn { enumext } { width-zero }
5374      {
5375        Invalid ~ '#1=#2' ~ \msg_line_context:.\\
5376        The ~ key ~ '#1'~ accepts ~ values  ~ > ~ 0pt.
5377      }
```

Messages used by show-length key in enumext.

```
5378  \msg_new:nnn { enumext } { list-lengths }
5379      {
5380        **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
5381        \__enumext_show_length:nnn { dim  } { labelsep     } {#1}
5382        \__enumext_show_length:nnn { dim  } { labelwidth   } {#1}
5383        \__enumext_show_length:nnn { dim  } { itemindent   } {#1}
5384        \__enumext_show_length:nnn { dim  } { leftmargin   } {#1}
5385        \__enumext_show_length:nnn { dim  } { rightmargin  } {#1}
5386        \__enumext_show_length:nnn { dim  } { listparindent } {#1}
5387        \__enumext_show_length:nnn { skip } { topsep     } {#1}
5388        \__enumext_show_length:nnn { skip } { parsep     } {#1}
5389        \__enumext_show_length:nnn { skip } { partopsep } {#1}
5390        \__enumext_show_length:nnn { skip } { itemsep    } {#1}
5391        ****************************************************
5392      }
```

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

```
5393  \msg_new:nnn { enumext } { list-lengths-not-nested }
5394      {
5395        **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
5396        \__enumext_show_length:nnn { dim  } { labelsep     } {#1}
5397        \__enumext_show_length:nnn { dim  } { labelwidth   } {#1}
5398        \__enumext_show_length:nnn { dim  } { itemindent   } {#1}
5399        \__enumext_show_length:nnn { dim  } { leftmargin   } {#1}
5400        \__enumext_show_length:nnn { dim  } { rightmargin  } {#1}
5401        \__enumext_show_length:nnn { dim  } { listparindent } {#1}
5402        \__enumext_show_length:nnn { skip } { topsep     } {#1}
5403        \__enumext_show_length:nnn { skip } { parsep     } {#1}
```

```
5404        \__enumext_show_length:nnn { skip } { partopsep } {#1}
5405        \__enumext_show_length:nnn { skip } { itemsep   } {#1}
5406        **************************************************
5407      }
```

Messages used by `ref` key.

```
5408  \msg_new:nnn { enumext } { key-ref-empty }
5409    {
5410      Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
5411    }
```

Messages used by `save-ans` key.

```
5412  \msg_new:nnn { enumext } { save-ans-empty }
5413    {
5414      Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
5415    }
5416  \msg_new:nnn { enumext } { save-ans-log }
5417    {
5418      * ~ Package ~ enumext: ~ Start ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5419    }
5420  \msg_new:nnn { enumext } { save-ans-log-hook }
5421    {
5422      * ~ Package ~ enumext: ~ Stop ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5423    }
5424  \msg_new:nnn { enumext } { save-ans-hook }
5425    {
5426      Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
5427    }
```

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

```
5428  \msg_new:nnn { enumext } { need-save-ans }
5429    {
5430      Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
5431    }
5432  \msg_new:nnn { enumext } { items-same-answer }
5433    {
5434      *****************************************\\
5435      * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~
5436      for ~ \c_left_brace_str #2 \c_right_brace_str\\
5437      * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~
5438      'OK', ~ all ~ items ~ with ~ answer.\\
5439      *****************************************
5440    }
5441  \msg_new:nnn { enumext } { item-greater-answer }
5442    {
5443      Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
5444      started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
5445      Items ~ > ~ Answers.
5446    }
5447  \msg_new:nnn { enumext } { item-less-answer }
5448    {
5449      Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
5450      started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
5451      Items ~ < ~ Answers.
5452    }
```

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

```
5453  \msg_new:nnn { enumext } { missing-starred }
5454    {
5455      Missing ~ '\c_backslash_str #1*' ~ #2.
5456    }
5457  \msg_new:nnn { enumext } { many-starred }
5458    {
5459      Many ~ '\c_backslash_str #1*' ~ #2.
5460    }
```

Messages used by `\printkeyans*` command.

```
5461  \msg_new:nnn { enumext } { print-starred }
5462    {
5463      \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
5464      #2 ~ environment ~  \msg_line_context:.
5465    }
```

Message for the nesting depth of the environment enumext.

```
5466  \msg_new:nnn { enumext } { list-too-deep }
5467    {
5468      Too ~ deep ~ nesting ~ for ~ 'enumext' ~ \msg_line_context:.~ \\
5469      The ~ maximum ~ level ~ of ~ nesting ~ is ~ 4.
5470    }
```

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

```
5471  \msg_new:nnn { enumext } { anskey-unnumber-item }
5472    {
5473      Can't ~ store ~ with ~ a ~ unnumbered ~ \c_backslash_str item ~ \msg_line_context:.
5474    }
5475  \msg_new:nnn { enumext } { anskey-already-stored }
5476    {
5477      Content ~ already ~ stored ~ for ~ this ~ \c_backslash_str item ~ \msg_line_context:.
5478    }
5479  \msg_new:nnn { enumext } { anskey-empty-arg }
5480    {
5481      Can't ~ store ~ empty ~ content ~ \msg_line_context:.
5482    }
5483  \msg_new:nnn { enumext } { anskey-wrong-place }
5484    {
5485      Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
5486      '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5487    }
5488  \msg_new:nnn { enumext } { anskey-nested }
5489    {
5490      The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
5491    }
5492  \msg_new:nnn { enumext } { anskey-math-mode }
5493    {
5494      #1 ~ can't ~ work ~ in ~ math ~ mode ~ \msg_line_context:.
5495    }
5496  \msg_new:nnn { enumext } { anskey-env-wrong }
5497    {
5498      The ~ environment ~ anskey* ~ cannot ~ use ~ in ~ '#1' ~ \msg_line_context:.
5499    }
5500  \msg_new:nnn { enumext } { anspic-wrong-place }
5501    {
5502      Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
5503      '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5504    }
5505  \msg_new:nnn { enumext } { command-wrong-place }
5506    {
5507      Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
5508      '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
5509    }
5510  \msg_new:nnnn { enumext } { anskey-env-key-unknown }
5511    {
5512      The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5513      'anskey*' ~ and ~ is ~ being ~ ignored.
5514    }
5515    {
5516      The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5517      Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5518    }
5519  \msg_new:nnnn { enumext } { anskey-env-key-value-unknown }
5520    {
5521      The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5522      'anskey*' ~ and ~ is ~ being ~ ignored.
5523    }
5524    {
5525      The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5526      Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5527    }
5528  \msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
5529    { The ~ key ~'#1'~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.}
5530    {
5531      The ~ command ~'\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5532      Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5533    }
```

```
5534 \msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
5535   { The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignor
5536   {
5537     The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5538     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5539   }
```

Messages used by keyans, keyans* and keyanspic environment.

```
5540 \msg_new:nnn { enumext } { keyans-nested }
5541   {
5542     The ~ environment ~ 'keyans' ~ can't ~ be  ~ nested  ~ \msg_line_context:.
5543   }
5544 \msg_new:nnn { enumext } { keyans-wrong-level }
5545   {
5546     Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:.~ \\
5547     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5548   }
5549 \msg_new:nnn { enumext } { wrong-place }
5550   {
5551     Wrong ~ place ~ for ~ '#1' ~ environment ~\msg_line_context:.~ \\
5552     '#1' ~ is ~ only ~ found ~ with ~ '#2' ~  in  ~  'enumext.
5553   }
5554 \msg_new:nnn { enumext } { keyanspic-nested }
5555   {
5556     The ~ environment ~ 'keyanspic' ~ can't ~ be  ~ nested~ \msg_line_context:.~.
5557   }
5558 \msg_new:nnn { enumext } { keyanspic-wrong-level }
5559   {
5560     Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:.~ \\
5561     The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5562   }
5563 \msg_new:nnn { enumext } { keyanspic-item-cmd }
5564   {
5565     Can't ~ use  ~ \c_backslash_str item ~ in ~ keyanspic ~ \msg_line_context:.
5566   }
5567 \msg_new:nnnn { enumext } { keyans-unknown-key }
5568   {
5569     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5570     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5571   }
5572   {
5573     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5574     ~ have ~ a ~ key ~ called ~'#1'.\\
5575     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5576   }
5577 \msg_new:nnnn { enumext } { keyans-unknown-key-value }
5578   {
5579     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5580     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5581   }
5582   {
5583     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5584     ~ have ~ a ~ key ~ called ~'#1'.\\
5585     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5586   }
```

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

```
5587 \msg_new:nnnn { enumext } { starred-unknown-key }
5588   {
5589     The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5590     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5591   }
5592   {
5593     The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5594     ~ have ~ a ~ key ~ called ~'#1'.\\
5595     Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5596   }
5597 \msg_new:nnnn { enumext } { starred-unknown-key-value }
5598   {
5599     The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5600     '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5601   }
```

```
5602     {
5603       The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5604       ~ have ~ a ~ key ~ called ~'#1'.\\
5605       Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5606     }
```

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

```
5607 \msg_new:nnnn { enumext } { standar-unknown-key }
5608     {
5609       The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_tl
5610       ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5611     }
5612     {
5613       The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5614       ~ have ~ a ~ key ~ called ~'#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5615       Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5616     }
5617 \msg_new:nnnn { enumext } { standar-unknown-key-value }
5618     {
5619       The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_t
5620       ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5621     }
5622     {
5623       The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5624       ~ have ~ a ~ key ~ called ~'#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5625       Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5626     }
```

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

```
5627 \msg_new:nnnn { enumext } { for-key-unknown }
5628     { The~key~'#1'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored.}
5629     {
5630       The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5631       Check~that~you~have~spelled~the~key~name~correctly.
5632     }
5633 \msg_new:nnnn { enumext } { for-key-value-unknown }
5634     { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored. }
5635     {
5636       The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5637       Check~that~you~have~spelled~the~key~name~correctly.
5638     }
```

Messages used by \getkeyans command.

```
5639 \msg_new:nnn { enumext } { undefined-storage-anskey }
5640     {
5641       Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
5642     }
```

Messages used by \miniright command.

```
5643 \msg_new:nnn { enumext } { missing-miniright }
5644     {
5645       Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\\
5646       The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
5647     }
5648 \msg_new:nnn { enumext } { wrong-miniright-place }
5649     {
5650       Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
5651       Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
5652     }
5653 \msg_new:nnn { enumext } { wrong-miniright-use }
5654     {
5655       Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
5656       '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
5657     }
5658 \msg_new:nnn { enumext } { wrong-miniright-starred }
5659     {
5660       Can't ~ use  ~ \c_backslash_str miniright ~ in ~ starred ~ environments ~ \msg_line_context:.
5661     }
5662 \msg_new:nnn { enumext } { many-miniright-used }
5663     {
5664       Can't ~ use  ~ \c_backslash_str miniright ~ more ~ than ~ once ~  \msg_line_context:.
5665     }
```

Messages used by `\setenumextmeta` command.

```
5666 \msg_new:nnn { enumext } { unknown-set }
5667   {
5668     Argument ~ [#1] ~ is ~ unknown ~ by ~  \c_backslash_str setenumextmeta ~ \msg_line_context:.
5669   }
5670 \msg_new:nnn { enumext } { already-defined }
5671   {
5672     The ~ key ~ '#1' ~ is ~ already ~ defined ~ \msg_line_context:.
5673   }
5674 \msg_new:nnn { enumext } { prohibited-unknown }
5675   {
5676     The ~ name ~ 'unknown' ~ can't ~ be ~ chosen~ for ~ a ~ meta ~ key ~ \msg_line_context:.
5677   }
```

Messages used by `enumext*` and `keyans*` environments.

```
5678 \msg_new:nnn { enumext } { nested }
5679   {
5680     The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ \msg_line_con
5681   }
5682 \msg_new:nnn { enumext } { nested-horizontal }
5683   {
5684     The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ in ~ '#1' ~ 
5685   }
5686 \msg_new:nnn { enumext } { item-joined }
5687   {
5688     Items ~ joined ~ (#1) ~ > ~ #2  ~ columns ~\msg_line_context:.
5689   }
5690 \msg_new:nnn { enumext } { item-joined-columns }
5691   {
5692     Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.
5693   }
```

12.51   Finish package

Finish package implementation.

```
5694 \file_input_stop:
5695 ⟨/package⟩
```

13   Index of Implementation

The italic numbers denote the pages where the corresponding entry is described, the numbers underlined and all others indicate the line on which they are implemented in the package code.