enumext
ENUMERATE EXERCISE SHEETS

V1.0   2024-11-10*

©2024 by Pablo González†

Abstract

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

Contents

Motivation and acknowledgments

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

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

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

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

---

License and Requirements

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

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

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

1 Introduction

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

   A) value   D) value
   B) value   E) value
   C) value

4. Question with image and label below:

A)   B)   C)

D)   E)
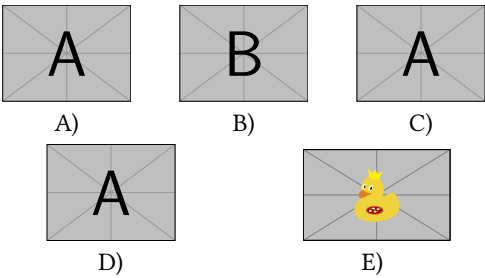
5. Question with image on right side:

A) value
B) value
C) value
D) correct
E) value

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

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

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

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

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

1.1 Description and usage

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

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

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

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

The package is loaded in the usual way:

```
\usepackage{enumext}
```

1.2 The concept of left margin

There is a direct relationship between the parameters \leftmargin, \itemindent, \labelwidth and \labelsep plus an *"extra space"* that makes it difficult to obtain the desired *horizontal spaces* in a list environment. Usually we don't want the list to go beyond the left margin of the page, but since these four values are related, that causes a problem.

The enumitem[6] package adds the \labelindent parameter to solve some of these problems. A simplified representation of this in the figure 1.
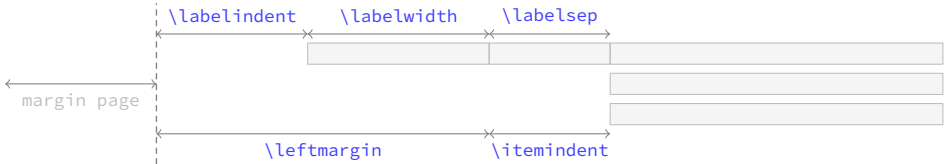
Figure 1: Representation of horizontal lengths in enumitem.

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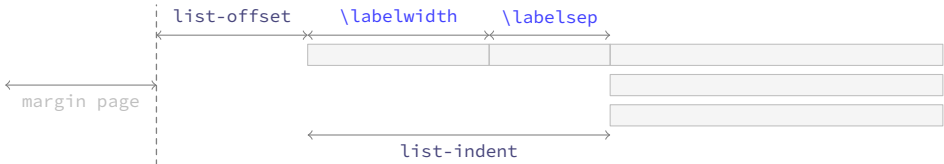
Figure 2: Representation of horizontal lengths concept in enumext.

In this way we reduce a *little* the amount of parameters we have to pass. With the default values of keys list-offset, list-indent, labelwidth and labelsep the lists will have the (usually) expected output for *"simple worksheets"*. The figure 3 shows the visual representation.
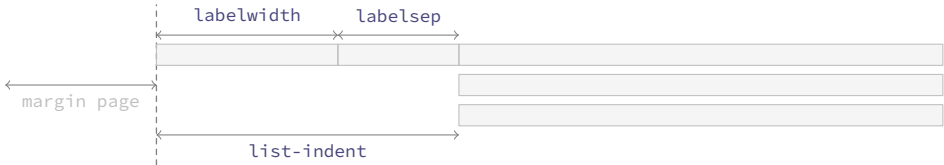
Figure 3: Default horizontal lengths list-offset=0pt, list-indent=\labelwidth+\labelsep in enumext.

1.3   User interface

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

1.3.1   Internal counters

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

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

1.3.2   Public dimension

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

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

1.3.3   Support for multicol

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

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

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

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

1.3.4   Support for `minipage`

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

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

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

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

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

1.3.6   Support for `\footnote`

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

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

The best way to solve this if *tagged* PDF is NOT active is to use Jean-François Burnol `footnotehyper`[9] package, it will support keeping the *"links"* if `hyperref` is loaded with the `hyperfootnotes=true` option (default). Load it is as follows:

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

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

2 The environments provided

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

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

Some considerations to take into account for this environment:

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

Example with `columns=2`

1. This text is in the first level.

X This text is in the first level.

2. This text is in the first level.

⋆ 4. This text is in the first level.

2.3 The command `\item*`

`\item*`

`\item*[`⟨*symbol*⟩`][`⟨*offset*⟩`]`

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

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

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

2.3.1   Keys for \item*

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

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

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

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

2.4   The command \item in enumext*

The \item command for the enumext* environment provides an *"first optional argument"* \item(⟨*columns*⟩) which *"joins items"* between columns. Let's consider the following examples adapted directly from the task package:

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

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

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

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

Z The tenth (89.28171pt)

3   The command \setenumext

\setenumext

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

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

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

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

4   The command \setenumextmeta

\setenumextmeta

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

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

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

5 The `keyval` system

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

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

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

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

5.1 Keys for `label` and `ref`

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

5.2 Keys for spaces

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

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

5.2.1 Vertical spaces

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

5.2.2 Horizontal spaces

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

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

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

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

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

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

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

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

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

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

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

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

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

5.3 Keys for add code

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

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

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

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

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

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

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

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

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

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

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

5.4 Keys for start, series and resume

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

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

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

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

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

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

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

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

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

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

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

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

5.5 Keys for multicols

columns = {⟨*integer*⟩} default: *1*

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

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

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

5.6 Keys for `minipage`

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

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

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

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

5.6.1 The command `\miniright`

`\miniright`

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

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

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

5.6.2 The key `mini-right`

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

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

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

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

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

6 The storage system

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

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

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

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

6.1 Keys for storage system

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

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

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

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

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

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

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

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

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

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

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

6.1.1   Keys for `label` and `ref`

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

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

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

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

6.1.2   Keys for `wrap` and `display`

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

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

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

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

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

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

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

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

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

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

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

`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.

6.1.3 Keys for debug and checking

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

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

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

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

6.2 The command `\anskey`

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

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

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

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

6.2.1 Keys for `\anskey`

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

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

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

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

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

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

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

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

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

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

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

Example

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

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

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

6.3 The environment anskey*

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

The environment `anskey*` takes a mandatory `{⟨body content⟩}` and *"stores it"* in the *sequence* and *prop list* `{⟨store name⟩}` set by `save-ans` key. If `save-ref` key are active and the `hyperref`[8] package is detected `\hyperlink` and `\hypertarget` will be used, otherwise the usual *"label and ref"* system provided by LaTeX will be used. By design the environment cannot be nested but full supports *"verbatim material"* in the body and it is assumed that each numbered `\item` or `\item*` within the environment in which it is active it has a *"single execution"* unless `\item` or `\item*` open a nested level or use the `no-store` key.

The `anskey*` environment is implemented using the `scontents` package, 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.

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

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

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

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

6.4.1 The \item* in keyans and keyans*

\item* `\item*`
`\item*[⟨content⟩]`

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

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

Example

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

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

  \item Text containing a question and image.

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

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

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

Some text

6.5 The environment `keyanspic`

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

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

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

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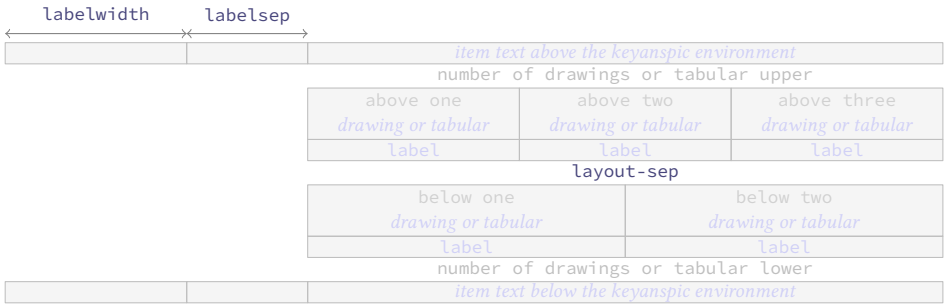
Figure 6: Representation of the `keyanspic` environment with `layout-sty`={⟨*3, 2*⟩} in enumext.

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

6.5.1 Keys for `keyanspic`

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

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

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

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

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

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

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

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

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

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

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

6.5.2 The command \anspic

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

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

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

Example

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

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

  \item Question with images and labels above.

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

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

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

1. Question with images and labels below.

A) B) C)

D) ∗ E) [note]

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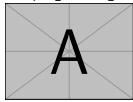
A) B) C)

D) ∗ E) [note]

3. Question with images and labels below on a single line.

A) B) C) D) ∗ E) [note]

🔖 Remember to pass the `alt={⟨description⟩}` key to the `\includegraphics` command when creating a *tagged* PDF.

6.6 Printing stored content

6.6.1 The command \getkeyans

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

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

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

6.6.2 The command \foreachkeyans

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

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

Options for command

sep = {⟨*code*⟩}   default: *{; }*

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

step = {⟨*integer*⟩}   default: *1*

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

start = {⟨*integer*⟩}   default: *1*

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

stop = {⟨*integer*⟩}   default: *0*

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

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

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

after = {⟨*code*⟩}   default: *empty*

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

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

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

6.6.3 The command \printkeyans

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

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

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

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

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

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

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

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

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

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

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

Example

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

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

  \item Related to Linux

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


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

\printkeyans{sample}
```

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

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

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.

---

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

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   ※

Example 3

A *"simple multiple choice"* test 📄.

1. First type of questions

   (A) value   (B) correct
   (C) value   (D) value

2. Second type of questions

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

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

3. Third type of questions

   (1) $2\alpha + 2\delta = 90°$
   (2) $\angle EDF = 45°$

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

4. Question with image and label below:

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

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

Test keys

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

Example 4

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

Factor $x^2 - 2x + 1$

Factor $3x + 3y + 3z$

The following questions need to be cuaqtified :)

True False

(a) $\alpha > \delta$
(b) $\LaTeX2e$ is cool?

Related to Linux

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

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

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

Example 5

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

1

Which choice best describes what happens in the passage?

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

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 Tagged PDF examples

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

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

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

🏷 All examples have been checked using `veraPDF` together with `ngpdf`.

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

9 The way of non-enumerated lists

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

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

Fake `itemize` environment

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

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

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

Fake `description` environment

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

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

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

SomeThing A short one-line description.
 This is an entry *without* a label.

Something A short *one-line* description text.

Something long A much *longer* description text may take more than one line or more than one paragraph. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

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

🔷 When *tagged* PDF is active the default `description` style is NOT available due to the redefinition of `\makelabel` for the `align` key which uses `\makebox` in this case, meaning that `\item[⟨content⟩]` will not extend beyond `\labelwidth` which causes overlaps,

Description indented by label

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

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

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

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

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

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

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

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

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

Description with multi-line labels

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

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

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

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

Final notes

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

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

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

Why list environments?

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

Why not random questions and other utilities

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

Why has it taken so long?

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

10   References

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

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

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

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

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

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

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

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

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

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

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

[12] The LATEX Project. "The LATEX 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.

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

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

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

11   Change history

V1.0 2024-11-10   – First public release.

12   Index of Documentation

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

13   Implementation

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

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

13.1   General conventions

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

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

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

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

13.2   Initial set up

Start the DocStrip guards.

```
1  ⟨*package⟩
```

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

```
2  ⟨@@=enumext⟩
```

13.3   Declaration of the package

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

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

Now declare the enumext package.

```
4  \ProvidesExplPackage {enumext} {2024-11-10} {1.0} {Enumerate exercise sheets}
```

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

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

13.4   Definition of variables

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

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

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

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

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

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

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

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

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

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

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

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

```
41 \cs_set_protected:Npn \__enumext_tmp:n #1
42   {
43     \tl_new:c { l__enumext_counter_#1_tl }
44   }
45 \clist_map_inline:nn { i, ii, iii, iv, v, vi, vii, viii } { \__enumext_tmp:n {#1} }
```

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

\c__enumext_counter_style_tl
\l__enumext_ref_key_arg_tl
\l__enumext_ref_the_count_tl
\l__enumext_the_counter_X_tl
\l__enumext_renew_the_count_X_tl

Internal variables used by ref key (§13.14).

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

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

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

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

```
57 \int_new:N  \g__enumext_resume_int
58 \int_new:N  \g__enumext_resume_vii_int
59 \tl_new:N   \l__enumext_resume_name_tl
60 \bool_new:N \l__enumext_resume_active_bool
61 \tl_new:N   \g__enumext_standar_series_tl
62 \tl_new:N   \g__enumext_starred_series_tl
```

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

\l__enumext_current_widest_dim
\g__enumext_counter_styles_tl
\g__enumext_widest_label_tl
\l__enumext_label_width_by_box

The variable \l__enumext_current_widest_dim stores the current label width, the variable \g__-enumext_counter_styles_tl stores the default ⟨*label style*⟩ and the variable \g__enumext_widest_-label_tl the label width. These variables are used by widest (§13.15) and label (§13.13) keys.

```
63 \dim_new:N \l__enumext_current_widest_dim
64 \tl_new:N  \g__enumext_counter_styles_tl
65 \tl_new:N  \g__enumext_widest_label_tl
66 \box_new:N \l__enumext_label_width_by_box
```

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

\l__enumext_leftmargin_tmp_X_bool
\l__enumext_leftmargin_tmp_X_dim
\l__enumext_leftmargin_X_dim
\l__enumext_itemindent_X_dim

The boolean variable \l__enumext_leftmargin_tmp_X_bool and the dimensional variable \l__enumext_-leftmargin_tmp_X_dim are used by the list-indent key (§13.18). The variables \l__enumext_-leftmargin_X_dim and \l__enumext_itemindent_X_dim are used and set by the function __enumext_-calc_hspace:NNNNNNNNNNNN (§13.38.1).

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

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

\l__enumext_multicols_above_X_skip
\l__enumext_multicols_below_X_skip
\g__enumext_multicols_right_X_skip
\l__enumext_align_label_pos_X_str

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

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

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

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

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

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

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

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

The bool vars \l__enumext_wrap_label_X_bool and \l__enumext_wrap_label_opt_X_bool are used by wrap-label and wrap-label* keys (§13.13), the integer \l__enumext_start_X_int are used by the start and start* keys (§13.15), the token list \l__enumext_fake_item_indent_X_tl is used by itemindent key (§13.18.1), the variables \l__enumext_label_fill_left_X_tl and \l__enumext_-label_fill_left_X_tl are used by the align key (§13.13). The boolean vars \l__enumext_vspace_-a_star_X_bool, \l__enumext_vspace_b_star_X_bool are used by above, above*, below and below* keys (§13.20).

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

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

The variable `\l__enumext_store_active_bool` setting by save-ans key (§13.26.1) activates all the mechanism related to `\anskey`, anskey*, keyans, keyans* and keyanspic environments.

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

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

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

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

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

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

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

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

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

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

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

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

Internal variables used by command `\printkeyans` (§13.47), show-pos, show-ans, mark-pos, mark-sep keys (§13.27), item-sym* key (§13.35), save-key key (§13.27.2) and *"storing structure"*.

```
126 \tl_new:N   \l__enumext_print_keyans_starred_tl
127 \bool_new:N \l__enumext_print_keyans_star_bool
128 \str_new:N  \l__enumext_mark_position_str
129 \dim_new:N  \l__enumext_mark_sep_tmpa_dim
130 \dim_new:N  \l__enumext_mark_sep_tmpb_dim
131 \int_new:N  \l__enumext_show_pos_tmp_int
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 (§13.42.1).

```
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_anspic_label_above_bool
146 \str_new:N  \l__enumext_anspic_mini_pos_str
```

```
147  \box_new:N  \l__enumext_anspic_label_box
148  \box_new:N  \l__enumext_anspic_body_box
149  \dim_new:N  \l__enumext_anspic_label_htdp_dim
150  \dim_new:N  \l__enumext_anspic_body_htdp_dim
```

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

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

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

```
151  \bool_new:N \l__enumext_check_answers_bool
152  \bool_new:N \g__enumext_check_ans_key_bool
153  \tl_new:N   \l__enumext_check_start_line_env_tl
154  \bool_new:N \l__enumext_item_wrap_key_bool
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.*)

\l__enumext_hyperref_bool
\l__enumext_footnotes_key_bool

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

```
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.)

\l__enumext_newlabel_arg_one_tl
\l__enumext_newlabel_arg_two_tl
\l__enumext_write_aux_file_tl
\l__enumext_label_copy_X_tl

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

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

```
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.*)

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

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

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

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

\l__enumext_item_starred_X_bool
l__enumext_item_column_pos_X_int
\g__enumext_item_count_all_X_int
\l__enumext_joined_item_X_int
\l__enumext_joined_item_aux_X_int
\l__enumext_tmpa_X_int
\l__enumext_tmpa_X_dim
\l__enumext_item_text_X_box
\l__enumext_joined_width_X_dim
\l__enumext_item_width_X_dim
\g__enumext_item_symbol_aux_X_tl
\l__enumext_align_label_X_str
\g__enumext_minipage_active_X_bool
\l__enumext_miniright_code_X_box
\g_enumext_minipage_center_X_bool

Internal variables used by enumext* and keyans* environments.

```
176  \cs_set_protected:Npn \__enumext_tmp:n #1
177    {
178      \bool_new:c { l__enumext_item_starred_#1_bool    }
179      \int_new:c  { l__enumext_item_column_pos_#1_int  }
180      \int_new:c  { g__enumext_item_count_all_#1_int   }
181      \int_new:c  { l__enumext_joined_item_#1_int      }
182      \int_new:c  { l__enumext_joined_item_aux_#1_int  }
183      \int_new:c  { l__enumext_tmpa_#1_int             }
184      \dim_new:c  { l__enumext_tmpa_#1_dim             }
185      \box_new:c  { l__enumext_item_text_#1_box        }
186      \dim_new:c  { l__enumext_joined_width_#1_dim     }
187      \dim_new:c  { l__enumext_item_width_#1_dim       }
188      \tl_new:c   { g__enumext_item_symbol_aux_#1_tl   }
```

```
189       \str_new:c  { l__enumext_align_label_#1_str      }
190       \bool_new:c { g__enumext_minipage_active_#1_bool }
191       \box_new:c  { l__enumext_miniright_code_#1_box   }
192       \bool_new:c { g__enumext_minipage_center_#1_bool }
193       \dim_new:c  { g__enumext_minipage_right_#1_dim   }
194       \skip_new:c { g__enumext_minipage_right_#1_skip  }
195     }
196 \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`.

```
197 \clist_const:Nn \c__enumext_all_envs_clist
198   {
199     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv},
200     {keyans}{v}, {enumext*}{vii}, {keyans*}{viii}
201   }
```

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

13.5 Some utility functions

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

```
202 \cs_generate_variant:Nn \keys_precompile:nnN { neN }
203 \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.

```
204 \cs_new_protected:Npn \__enumext_at_begin_document:n #1
205   {
206     \hook_gput_code:nnn {begindocument} {enumext} { #1 }
207   }
```

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

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

```
208 \cs_new_protected:Npn \__enumext_after_env:nn #1 #2
209   {
210     \hook_gput_code:nnn {env/#1/after} {enumext} {#2}
211   }
212 \cs_new_protected:Npn \__enumext_before_env:nn #1 #2
213   {
214     \hook_gput_code:nnn {env/#1/before} {enumext} {#2}
215   }
```

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

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

```
216 \cs_new:Nn \__enumext_level:
217   {
218     \int_to_roman:n { \l__enumext_level_int }
219   }
```

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

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

```
220 \prg_new_protected_conditional:Npnn \__enumext_if_is_int:n #1 { T, F, TF }
221   {
222     \regex_match:nnTF { ^[\+\-]?[\d]+$ } {#1} % $
223       { \prg_return_true: }
224       { \prg_return_false: }
225   }
```

(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.

```
226  \cs_new_protected:Nn \__enumext_regex_counter_style:
227    {
228      \tl_map_inline:Nn \c__enumext_counter_style_tl
229        {
230          \regex_replace_once:nnN { \c{##1}\* }
231            { \c{##1}\cB{\u{l__enumext_ref_the_count_tl}\cE} } \l__enumext_ref_key_arg_tl
232        }
233    }
```

(*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.

```
234  \cs_new:Npn \__enumext_show_length:nnn #1 #2 #3
235    {
236      * ~ #2
237      \prg_replicate:nn { 14 - \str_count:n {#2} } { ~ }
238        = ~ \use:c { #1_use:c } { l__enumext_#2_#3_#1 } \\
239    }
```

(*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.

```
240  \cs_new_protected:Nn \__enumext_unskip_unkern:
241    {
242      \int_case:nnT { \lastnodetype }
243        {
244          { 11 }{ \unskip }
245          { 12 }{ \unkern }
246        }
247    }
```

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

13.5.1 Utilities for environments and levels

__enumext_is_not_nested:
__enumext_is_on_first_level:

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

```
248  \cs_new_protected:Nn \__enumext_is_not_nested:
249    {
250      \str_case:en { \@currenvir }
251        {
252          {enumext}
253            {
254              \tl_set:Nn \l__enumext_envir_name_tl { enumext }
255              \bool_lazy_and:nnT
256                { \bool_not_p:n { \g__enumext_standar_bool } }
257                { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
258                {
259                  \bool_gset_true:N \g__enumext_standar_bool
260                }
261            }
262          {enumext*}
263            {
264              \tl_set:Nn \l__enumext_envir_name_tl { enumext* }
265              \bool_lazy_and:nnT
266                { \bool_not_p:n { \g__enumext_starred_bool } }
267                { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
268                {
269                  \bool_gset_true:N \g__enumext_starred_bool
270                }
271            }
272        }
273    }
```

The function __enumext_is_on_first_level: will set the variables \l__enumext_standar_first_-bool (§13.26.1), \l__enumext_starred_first_bool (§13.26.1) and \l__enumext_anskey_env_bool (§13.31) to *"true"* only if the environment is not nested and we are in the *"first level"* of it . We will also save the *start line number* of each environment in the variable \g__enumext_start_line_tl and the *name* of each environment in the variable \g__enumext_envir_name_tl to use in messages related to the check-ans key and .log file.

```
274 \cs_new_protected:Nn \__enumext_is_on_first_level:
275   {
276     \bool_lazy_all:nT
277       {
278         { \bool_if_p:N \g__enumext_standar_bool }
279         { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
280         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
281       }
282       {
283         \bool_set_true:N \l__enumext_standar_first_bool
284         \bool_set_true:N \l__enumext_anskey_env_bool
285         \tl_gset:Nn \g__enumext_envir_name_tl { enumext }
286         \tl_gset:Ne \g__enumext_start_line_tl
287           {
288             on ~ line ~ \exp_not:V \inputlineno
289           }
290       }
291     \bool_lazy_all:nT
292       {
293         { \bool_if_p:N \g__enumext_starred_bool }
294         { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
295         { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
296       }
297       {
298         \bool_set_true:N \l__enumext_starred_first_bool
299         \bool_set_true:N \l__enumext_anskey_env_bool
300         \tl_gset:Nn \g__enumext_envir_name_tl { enumext* }
301         \tl_gset:Ne \g__enumext_start_line_tl
302           {
303             on ~ line ~ \exp_not:V \inputlineno
304           }
305       }
306   }
```

(*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.

```
307 \cs_new_protected:Nn \__enumext_keyans_name_and_start:
308   {
309     \str_case:en { \@currenvir }
310       {
311         {keyans}
312           {
313             \tl_set:Nn \l__enumext_envir_name_tl { keyans }
314             \tl_set:Ne \l__enumext_check_start_line_env_tl
315               {
316                 in ~ 'keyans' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
317               }
318           }
319         {keyans*}
320           {
321             \tl_set:Nn \l__enumext_envir_name_tl { keyans* }
322             \tl_set:Ne \l__enumext_check_start_line_env_tl
323               {
324                 in ~ 'keyans*' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
325               }
326           }
327         {keyanspic}
328           {
329             \tl_set:Nn \l__enumext_envir_name_tl { keyanspic }
330             \tl_set:Ne \l__enumext_check_start_line_env_tl
331               {
```

```
332                    in ~ 'keyanspic' ~ start ~ on ~ line ~ \exp_not:V \inputlineno
333                }
334            }
335        }
336    }
```

(*End of definition for* __enumext_keyans_name_and_start:.)

13.5.2   Utilities for log and terminal

__enumext_reset_global_vars:
__enumext_reset_global_int:
__enumext_reset_global_bool:
__enumext_reset_global_tl:

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

```
337  \cs_new_protected:Nn \__enumext_reset_global_vars:
338    {
339      \__enumext_reset_global_int:
340      \__enumext_reset_global_bool:
341      \__enumext_reset_global_tl:
342    }
343  \cs_new_protected:Nn \__enumext_reset_global_int:
344    {
345      \int_gzero:N \g__enumext_item_number_int
346      \int_gzero:N \g__enumext_item_anskey_int
347      \int_gzero:N \g__enumext_item_answer_diff_int
348    }
349  \cs_new_protected:Nn \__enumext_reset_global_bool:
350    {
351      \bool_gset_false:N \g__enumext_check_ans_key_bool
352      \bool_gset_false:N \g__enumext_standar_bool
353      \bool_gset_false:N \g__enumext_starred_bool
354    }
355  \cs_new_protected:Nn \__enumext_reset_global_tl:
356    {
357      \tl_gclear:N \g__enumext_store_name_tl
358      \tl_gclear:N \g__enumext_start_line_tl
359      \tl_gclear:N \g__enumext_envir_name_tl
360    }
```

(*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.

```
361  \cs_new_protected:Nn \__enumext_log_global_vars:
362    {
363      \msg_log:nneeee { enumext } { prop-seq-int-hook }
364        { \g__enumext_store_name_tl }
365        { \prop_count:c { g__enumext_ \g__enumext_store_name_tl _prop } }
366        { \seq_count:c { g__enumext_ \g__enumext_store_name_tl _seq } }
367        { \int_use:c { g__enumext_resume_ \g__enumext_store_name_tl _int } }
368    }
```

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.

```
369  \cs_new_protected:Nn \__enumext_log_answer_vars:
370    {
371      \msg_log:nneee { enumext } { item-answer-hook }
372        { \int_use:N \g__enumext_item_number_int }
373        { \int_use:N \g__enumext_item_anskey_int }
374        { \int_eval:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int} }
375    }
```

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

13.6   Copying list and minipage environments

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

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

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

$\minipage[\langle pos\rangle][\langle height\rangle][\langle inner\text{-}pos\rangle]\{\langle width\rangle\}$
$\quad\langle internal\ implement\rangle$
\endminipage

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

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

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

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

```
376  \__enumext_at_begin_document:n
377    {
378      \cs_new_eq:NN    \__enumext_start_list:nn \list
379      \cs_new_eq:NN    \__enumext_stop_list: \endlist
380      \NewCommandCopy \__enumext_item_std:w \item
381    }
```

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

```
382  \__enumext_at_begin_document:n
383    {
384      \cs_new_eq:NN \__enumext_minipage:w \minipage
385      \cs_new_eq:NN \__enumext_endminipage: \endminipage
386    }
```

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

13.7   Compatibility with hyperref and footnotehyper

__enumext_after_hyperref:
__enumext_hypertarget:nn
__enumext_phantomsection:

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

```
387  \hook_gput_code:nnn { begindocument } { enumext } { \__enumext_after_hyperref: }
388  \hook_gset_rule:nnnn { begindocument } { enumext } { after } { hyperref }
```

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

```
389  \cs_new_protected:Nn \__enumext_after_hyperref:
390    {
391      \IfPackageLoadedTF { hyperref }
392        {
393          \msg_info:nnn { enumext } { package-load } { hyperref }
394          \bool_set_true:N \l__enumext_hyperref_bool
395          \IfHyperBoolean{hyperfootnotes}
396            {
397              \bool_set_true:N \l__enumext_footnotes_key_bool
398            }
399            {  }
400        }
401        {  }
```

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.

```
402      \bool_if:NT \l__enumext_footnotes_key_bool
403        {
404          \IfPackageLoadedTF { footnotehyper }
405            {
406              \msg_info:nnn { enumext } { package-load } { footnotehyper }
407            }
408            {
409              \bool_set_false:N \l__enumext_footnotes_key_bool
410            }
411        }
```

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.

```
412      \bool_if:NTF \l__enumext_hyperref_bool
413        {
```

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

(*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.

```
422  \cs_new_protected:Npn \__enumext_newlabel:nn #1 #2
423    {
424      \protected@write \@auxout { }
425        {
426          \token_to_str:N \newlabel {#1}
427            {
428              {#2}
429              \bool_if:NT \l__enumext_hyperref_bool
430                { { \thepage } {#2} {#1} }
431              { }
432            }
433        }
434      \__enumext_hypertarget:nn {#1} { }
435      \__enumext_phantomsection:
436    }
```

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

13.8   Internal redefining \footnote **command**

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

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

```
437  \cs_new_protected:Nn \__enumext_footnotetext:nn
438    {
439      \footnotetext[#1]{#2}
440    }
441  \cs_new_protected:Nn \__enumext_renew_footnote:
442    {
443      \RenewDocumentCommand \footnote { o +m }
444        {
445          \tl_if_novalue:nTF {##1}
446            {
447              \stepcounter{footnote}
448              \int_gset_eq:Nc \g__enumext_footnote_standar_int { c@footnote }
449            }
450            {
451              \int_gset:Nn \g__enumext_footnote_standar_int { ##1 }
452            }
453          \footnotemark [ \g__enumext_footnote_standar_int ]
454          \seq_gput_right:Nn \g__enumext_footnote_standar_arg_seq { ##2 }
455          \seq_gput_right:NV
456            \g__enumext_footnote_standar_int_seq \g__enumext_footnote_standar_int
457        }
458    }
459  \cs_new_protected:Nn \__enumext_print_footnote:
460    {
461      \seq_if_empty:NF \g__enumext_footnote_standar_int_seq
462        {
463          \seq_map_pairwise_function:NNN
```

```
464              \g__enumext_footnote_standar_int_seq
465              \g__enumext_footnote_standar_arg_seq
466              \__enumext_footnotetext:nn
467          }
468      \seq_gclear:N \g__enumext_footnote_standar_arg_seq
469      \seq_gclear:N \g__enumext_footnote_standar_int_seq
470  }
```

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

```
471  \cs_new_protected:Nn \__enumext_renew_footnote_mini:
472  {
473      \RenewDocumentCommand \footnote { o +m }
474          {
475          \tl_if_novalue:nTF {##1}
476              {
477                  \stepcounter{footnote}
478                  \int_gset_eq:Nc \g__enumext_footnote_starred_int { c@footnote }
479              }
480              {
481                  \int_gset:Nn \g__enumext_footnote_starred_int { ##1 }
482              }
483          \footnotemark [ \g__enumext_footnote_starred_int ]
484          \seq_gput_right:Nn \g__enumext_footnote_starred_arg_seq { ##2 }
485          \seq_gput_right:NV
486              \g__enumext_footnote_starred_int_seq \g__enumext_footnote_starred_int
487          }
488  }
489  \cs_new_protected:Nn \__enumext_print_footnote_mini:
490  {
491      \seq_if_empty:NF \g__enumext_footnote_starred_int_seq
492          {
493          \seq_map_pairwise_function:NNN
494              \g__enumext_footnote_starred_int_seq
495              \g__enumext_footnote_starred_arg_seq
496              \__enumext_footnotetext:nn
497          }
498      \seq_gclear:N \g__enumext_footnote_starred_arg_seq
499      \seq_gclear:N \g__enumext_footnote_starred_int_seq
500  }
```

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

__enumext_renew_footnote_standar:
__enumext_print_footnote_standar:
__enumext_renew_footnote_starred:
__enumext_print_footnote_starred:

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

```
501  \cs_new_protected:Nn \__enumext_renew_footnote_standar:
502  {
503      \bool_if:NT \g__enumext_standar_bool
504          {
505          \IfDocumentMetadataTF
506              {
507                  \__enumext_renew_footnote:
508              }
509              {
510                  \bool_if:NF \l__enumext_footnotes_key_bool
511                      {
512                          \__enumext_renew_footnote:
513                      }
514              }
515          }
516  }
517  \cs_new_protected:Nn \__enumext_print_footnote_standar:
518  {
519      \bool_if:NT \g__enumext_standar_bool
520          {
521          \IfDocumentMetadataTF
522              {
523                  \__enumext_print_footnote:
524              }
525              {
```

```
526                 \bool_if:NF \l__enumext_footnotes_key_bool
527                   {
528                     \__enumext_print_footnote:
529                   }
530               }
531           }
532       }
```

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

```
533   \cs_new_protected:Nn \__enumext_renew_footnote_starred:
534     {
535       \IfDocumentMetadataTF
536         {
537           \__enumext_renew_footnote_mini:
538         }
539         {
540           \bool_if:NF \l__enumext_footnotes_key_bool
541             {
542               \__enumext_renew_footnote_mini:
543             }
544         }
545     }
546   \cs_new_protected:Nn \__enumext_print_footnote_starred:
547     {
548       \IfDocumentMetadataTF
549         {
550           \__enumext_print_footnote_mini:
551         }
552         {
553           \bool_if:NF \l__enumext_footnotes_key_bool
554             {
555               \__enumext_print_footnote_mini:
556             }
557         }
558     }
```

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

```
559   \__enumext_after_env:nn { enumext* }
560     {
561       \__enumext_print_footnote_starred:
562     }
563   \__enumext_after_env:nn { keyans* }
564     {
565       \__enumext_print_footnote_starred:
566     }
```

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

13.9   The internal minipage environment

__enumext_internal_mini_page:
__enumext_mini_env*

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

```
567   \cs_new_protected:Nn \__enumext_internal_mini_page:
568     {
569       \int_compare:nNnT { \l__enumext_level_int } = { 0 }
570         {
571           \DeclareDocumentEnvironment{__enumext_mini_page}{ m }
572             {
573               \__enumext_renew_footnote_standar:
574               \__enumext_minipage:w [ t ] { ##1 }
575                 \legacy_if_gset_false:n { @minipage }
576                 \skip_vertical:N \c_zero_skip
577             }
```

```
578            {
579                \skip_vertical:N \c_zero_skip
580              \__enumext_endminipage:
581              \__enumext_print_footnote_standar:
582            }
583          }
584      }
```

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

13.10   Definition of public dimension

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

```
585 \dim_zero_new:N \itemwidth
```

13.11   Definition of counters

__enumext_define_counters:Nn
enumXi
enumXii
enumXiii
enumXiv
enumXv
enumXvi
enumXvii
enumXviii

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

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

#2 :   The counter's name.

```
586 \cs_new_protected:Npn \__enumext_define_counters:Nn #1 #2
587   {
588     \cs_if_exist:cTF { c@ #2 }
589       { \msg_fatal:nnn { enumext } { counters }{ #2 } }
590       {
591         \tl_set:Nn #1 { #2 }
592         \newcounter { #2 }
593       }
594   }
```

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.

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

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

13.12   Definition of labels

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

__enumext_register_counter_style:Nn

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

```
603 \cs_new_protected:Npn \__enumext_register_counter_style:Nn #1 #2
604   {
605     \tl_const:cn { c__enumext_widest_ \cs_to_str:N #1 _tl } {#2}
606     \tl_gput_right:Nn \g__enumext_counter_styles_tl {#1}
607   }
608 \__enumext_register_counter_style:Nn \arabic { 0 }
609 \__enumext_register_counter_style:Nn \Alph   { M }
610 \__enumext_register_counter_style:Nn \alph   { m }
611 \__enumext_register_counter_style:Nn \Roman  { VIII }
612 \__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.

```
613 \cs_new_protected:Npn \__enumext_label_width_by_box:Nn #1 #2
614   {
615     \hbox_set:Nn \l__enumext_label_width_by_box {#2}
616     \dim_set:Nn #1 { \box_wd:N \l__enumext_label_width_by_box }
617   }
618 \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.

```
619 \cs_new_protected:Npn \__enumext_label_style:Nnn #1 #2 #3
620   {
621     \tl_clear_new:N #1
622     \tl_put_right:Ne #1 { \tl_trim_spaces:n {#3} }
623     \tl_gset_eq:NN \g__enumext_widest_label_tl #1
624     \tl_map_inline:Nn \g__enumext_counter_styles_tl
625       {
626         \tl_replace_all:Nne #1 { ##1* } { \exp_not:N ##1 {#2} }
627         \tl_greplace_all:Nne \g__enumext_widest_label_tl { ##1* }
628           { \tl_use:c { c__enumext_widest_ \cs_to_str:N ##1 _tl } }
629       }
630     \__enumext_label_width_by_box:Nn \l__enumext_current_widest_dim
631       { \tl_use:N \g__enumext_widest_label_tl }
632     \tl_set_eq:cN { the #2 } #1
633   }
634 \cs_generate_variant:Nn \__enumext_label_style:Nnn { cvn }
```

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

13.13 Setting keys associated with label

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

mode-box

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

```
635 \cs_set_protected:Npn \__enumext_tmp:n #1
636   {
637     \keys_define:nn { enumext /  #1 }
638       {
639         mode-box .bool_set:N = \l__enumext_mode_box_bool,
640         mode-box .initial:n  = false,
641         mode-box .value_forbidden:n = true,
642       }
643   }
644 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

font
labelsep
labelwidth
wrap-label
wrap-label*

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

```
645 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
646   {
647     \keys_define:nn { enumext / #1 }
648       {
649         font       .tl_set:c   = { l__enumext_label_font_style_#2_tl },
650         font       .value_required:n = true,
651         labelsep   .dim_set:c  = { l__enumext_labelsep_#2_dim },
652         labelsep   .initial:n  = {0.3333em},
653         labelsep   .value_required:n = true,
654         labelwidth .dim_set:c  = { l__enumext_labelwidth_#2_dim },
655         labelwidth .value_required:n = true,
656         wrap-label .cs_set_protected:cp = { __enumext_wrapper_label_#2:n } ##1,
657         wrap-label .initial:n  = {##1},
658         wrap-label .value_required:n = true,
```

```
659          wrap-label* .code:n = {
660                              \bool_set_true:c { l__enumext_wrap_label_opt_#2_bool }
661                              \keys_set:nn { enumext / #1 } { wrap-label = {##1} }
662                          },
663          wrap-label* .value_required:n = true,
664      }
665    }
666 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

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

```
667 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
668   {
669     \keys_define:nn { enumext / #1 }
670       {
671         align .choice:,
672         align / left     .code:n =
673                         {
674                             \tl_clear:c { l__enumext_label_fill_left_#2_tl  }
675                             \tl_set:cn  { l__enumext_label_fill_right_#2_tl } { \hfill }
676                             \str_set:cn { l__enumext_align_label_pos_#2_str } { l }
677                         },
678         align / right    .code:n =
679                         {
680                             \tl_set:cn  { l__enumext_label_fill_left_#2_tl  } { \hfill }
681                             \tl_clear:c { l__enumext_label_fill_right_#2_tl }
682                             \str_set:cn { l__enumext_align_label_pos_#2_str } { r }
683                         },
684         align / center   .code:n =
685                         {
686                             \tl_set:cn { l__enumext_label_fill_left_#2_tl  } { \hfill }
687                             \tl_set:cn { l__enumext_label_fill_right_#2_tl } { \hfill }
688                             \str_set:cn { l__enumext_align_label_pos_#2_str } { c }
689                         },
690         align / unknown .code:n =
691                         \msg_error:nneee { enumext } { unknown-choice }
692                         { align } { left, ~ right, ~  center } { \exp_not:n {##1} },
693         align .initial:n  = left,
694         align .value_required:n  = true,
695       }
696   }
697 \clist_map_inline:nn
698   {
699     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
700   }
701   { \__enumext_tmp:nn #1 }

702 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
703   {
704     \keys_define:nn { enumext / #1 }
705       {
706         align .choice:,
707         align / left     .code:n = \str_set:cn { l__enumext_align_label_#2_str } { l },
708         align / right    .code:n = \str_set:cn { l__enumext_align_label_#2_str } { r },
709         align / center   .code:n = \str_set:cn { l__enumext_align_label_#2_str } { c },
710         align / unknown .code:n =
711                         \msg_error:nneee { enumext } { unknown-choice }
712                         { align } { left, ~ right, ~  center } { \exp_not:n {##1} },
713         align .initial:n = left,
714         align .value_required:n = true,
715       }
716   }
717 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }
```

(*End of definition for* align.)

13.14   Setting label and ref keys

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

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

label
ref
\l__enumext_label_i_tl
\l__enumext_label_ii_tl
\l__enumext_label_iii_tl
\l__enumext_label_iv_tl

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

```
718  \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
719    {
720      \keys_define:nn { enumext / #1 }
721        {
722          label .code:n     = {
723                             \__enumext_label_style:cvn { l__enumext_label_#2_tl }
724                               { l__enumext_counter_#2_tl } {##1}
725                             \dim_set_eq:cN  { l__enumext_labelwidth_#2_dim }
726                               \l__enumext_current_widest_dim
727
728                             },
729          label .initial:n = #3,
730          label .value_required:n = true,
731          ref    .code:n     = \__enumext_standar_ref:n {##1},
732          ref    .value_required:n = true,
733        }
734    }
735  \__enumext_tmp:nnn { level-1 } {   i } { \arabic*.}
736  \__enumext_tmp:nnn { level-2 } {  ii } { (\alph*) }
737  \__enumext_tmp:nnn { level-3 } { iii } { \roman*. }
738  \__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.

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

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

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

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

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

label
ref
\l__enumext_label_vii_tl
\l__enumext_label_viii_tl

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

```
765  \cs_set_protected:Npn \__enumext_tmp:nnn #1 #2 #3
766    {
```

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

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

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

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

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.

```
819 \cs_new_protected:Nn \__enumext_starred_ref:
820   {
821     \int_compare:nNnT { \l__enumext_level_h_int } = { 1 }
822       {
823         \tl_if_empty:NF \l__enumext_renew_the_count_vii_tl
824           {
825             \tl_use:N \l__enumext_renew_the_count_vii_tl
826           }
827       }
828     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
829       {
830         \tl_if_empty:NF \l__enumext_renew_the_count_viii_tl
831           {
```

```
832                       \tl_use:N \l__enumext_renew_the_count_viii_tl
833                     }
834                   }
835             }
```

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

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

label
ref
\l__enumext_label_v_tl
\l__enumext_label_vi_tl

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

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

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

__enumext_keyans_ref:n
__enumext_keyans_ref:

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

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

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

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

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

13.15   Setting `start`, `start*` and `widest` keys

__enumext_start_from:NNn
__enumext_start_from:ccn
__enumext_start_from:cce

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

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

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

```
875  \cs_new_protected:Npn \__enumext_start_from:NNn #1 #2 #3
876    {
```

```
877        \__enumext_if_is_int:nTF { #3 }
878          {
879            \int_set:Nn #2 {#3}
880          }
881          {
882            \regex_match:nVT { \c{Alph} | \c{alph} } {#1}
883              { \int_set:Nn #2 { \int_from_alph:n {#3} } }
884            \regex_match:nVT { \c{Roman} | \c{roman} } {#1}
885              { \int_set:Nn #2  { \int_from_roman:n {#3} } }
886          }
887      }
888  \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.

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

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

start   Now define and set start*, start and widest keys for enumext, enumext*, keyans and keyans* environ-
start*   ments.
widest
```
905  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
906    {
907      \keys_define:nn { enumext / #1 }
908        {
909          start* .code:n     = {
910                                 \__enumext_start_from:ccn
911                                   { l__enumext_label_#2_tl }
912                                   { l__enumext_start_#2_int } {##1}
913                               },
914          start* .value_required:n = true,
915          start  .code:n     = {
916                                 \__enumext_start_from:cce
917                                   { l__enumext_label_#2_tl }
918                                   { l__enumext_start_#2_int } { \int_eval:n {##1} }
919                               },
920          start  .initial:n = 1,
921          start  .value_required:n = true,
922          widest .code:n     = {
923                                 \__enumext_widest_from:nccn {#2}
924                                   { l__enumext_label_#2_tl }
925                                   { l__enumext_labelwidth_#2_dim } {##1}
926                               },
927          widest .value_required:n = true,
928        }
929    }
930  \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

13.16 Setting keys for vertical spaces

topsep
partopsep
parsep
noitemsep
nosep

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

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

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

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

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

13.17 Setting base-fix key

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

base-fix
__enumext_nested_base_line_fix:

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

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

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

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

```
981  \cs_new_protected:Nn \__enumext_nested_base_line_fix:
982    {
983      \bool_lazy_all:nT
984        {
985          { \bool_if_p:N \l__enumext_starred_first_bool }
986          { \bool_if_p:N \l__enumext_base_line_fix_bool }
987          { \bool_not_p:n { \l__enumext_print_keyans_star_bool } }
988        }
989        {
990          \mode_leave_vertical:
991          \vspace { -\dim_eval:n { \baselineskip + \parsep } }
992        }
```

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

```
993      \bool_lazy_and:nnT
994        { \bool_if_p:N \l__enumext_starred_first_bool }
995        { \bool_if_p:N \l__enumext_print_keyans_star_bool }
996        {
997          \mode_leave_vertical:
998          \skip_vertical:n { -\baselineskip }
999          \skip_vertical:N \c_zero_skip
1000       }
```

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

```
1001     \keys_set:nn { enumext / level-1 }
1002       {
1003         topsep = 0pt, above = 0pt, above* = 0pt,
1004       }
1005     \bool_set_false:N \l__enumext_base_line_fix_bool
1006   }
```

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

13.18 Setting keys for horizontal spaces

itemindent
rightmargin
listparindent
list-offset
list-indent

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

```
1007  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1008    {
1009      \keys_define:nn { enumext / #1 }
1010        {
1011          itemindent    .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
1012          itemindent    .value_required:n = true,
1013          rightmargin   .dim_set:c = { l__enumext_rightmargin_#2_dim },
1014          rightmargin   .value_required:n = true,
1015          listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
1016          listparindent .value_required:n = true,
1017          list-offset   .dim_set:c = { l__enumext_listoffset_#2_dim },
1018          list-offset   .value_required:n = true,
1019          list-indent   .code:n    =
1020                             \bool_set_true:c { l__enumext_leftmargin_tmp_#2_bool }
1021                             \dim_set:cn { l__enumext_leftmargin_tmp_#2_dim } {##1},
1022          list-indent   .value_required:n = true,
1023        }
1024    }
1025  \clist_map_inline:nn
1026    {
1027      {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {keyans}{v}
1028    }
1029    { \__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.

```
1030 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1031   {
1032     \keys_define:nn { enumext / #1 }
1033       {
1034         itemindent     .dim_set:c = { l__enumext_fake_item_indent_#2_dim },
1035         itemindent     .value_required:n = true,
1036         rightmargin    .dim_set:c = { l__enumext_rightmargin_#2_dim },
1037         rightmargin    .value_required:n = true,
1038         listparindent .dim_set:c = { l__enumext_listparindent_#2_dim },
1039         listparindent .value_required:n = true,
1040         list-offset    .dim_set:c = { l__enumext_listoffset_#2_dim },
1041         list-offset    .value_required:n = true,
1042         list-indent    .meta:n     = { list-offset = ##1 },
1043         list-indent    .value_required:n = true,
1044       }
1045   }
1046 \clist_map_inline:nn
1047   {
1048     {enumext*}{vii}, {keyans*}{viii}
1049   }
1050   { \__enumext_tmp:nn #1 }
```

13.18.1   Functions for setting the fake itemindent

__enumext_fake_item_indent:
__enumext_keyans_fake_item_indent:
__enumext_fake_item_indent_vii:
__enumext_fake_item_indent_viii:

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

```
1051 \cs_set_protected:Nn \__enumext_fake_item_indent:
1052   {
1053     \dim_compare:nNnT
1054       { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
1055       >
1056       { \c_zero_dim }
1057       {
1058         \tl_set:ce { l__enumext_fake_item_indent_ \__enumext_level: _tl }
1059           {
1060             \exp_not:N \mode_leave_vertical:
1061             \exp_not:n { \skip_horizontal:n }
1062               { \dim_use:c { l__enumext_fake_item_indent_ \__enumext_level: _dim } }
1063             \exp_not:N \ignorespaces
1064           }
1065       }
1066   }
1067 \cs_set_protected:Nn \__enumext_keyans_fake_item_indent:
1068   {
1069     \dim_compare:nNnT
1070       { \l__enumext_fake_item_indent_v_dim } > { \c_zero_dim }
1071       {
1072         \tl_set:Ne \l__enumext_fake_item_indent_v_tl
1073           {
1074             \exp_not:N \mode_leave_vertical:
1075             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_v_dim
1076             \exp_not:N \ignorespaces
1077           }
1078       }
1079   }
1080 \cs_set_protected:Nn \__enumext_fake_item_indent_vii:
1081   {
1082     \dim_compare:nNnT
1083       { \l__enumext_fake_item_indent_vii_dim } > { \c_zero_dim }
1084       {
1085         \tl_set:Ne \l__enumext_fake_item_indent_vii_tl
1086           {
1087             \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_vii_dim
1088             \exp_not:N \ignorespaces
1089           }
1090       }
```

```
1091        }
1092  \cs_set_protected:Nn \__enumext_fake_item_indent_viii:
1093    {
1094      \dim_compare:nNnT
1095        { \l__enumext_fake_item_indent_viii_dim } > { \c_zero_dim }
1096        {
1097          \tl_set:Ne \l__enumext_fake_item_indent_viii_tl
1098            {
1099              \exp_not:N \skip_horizontal:N \l__enumext_fake_item_indent_viii_dim
1100              \exp_not:N \ignorespaces
1101            }
1102        }
1103    }
```

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

13.19   Setting show-length key

show-length

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

```
1104  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1105    {
1106      \keys_define:nn { enumext / #1 }
1107        {
1108          show-length .bool_set:c = { l__enumext_show_length_#2_bool },
1109          show-length .initial:n  = false,
1110        }
1111    }
1112  \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

13.20   Setting before, after and first keys

before
before*
after
first

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

```
1113  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1114    {
1115      \keys_define:nn { enumext / #1 }
1116        {
1117          before  .tl_set:c   = { l__enumext_before_no_starred_key_#2_tl },
1118          before  .value_required:n = true,
1119          before* .tl_set:c   = { l__enumext_before_starred_key_#2_tl },
1120          before* .value_required:n = true,
1121          after   .tl_set:c   = { l__enumext_after_stop_list_#2_tl },
1122          after   .value_required:n = true,
1123          first   .tl_set:c   = { l__enumext_after_list_args_#2_tl },
1124          first   .value_required:n = true,
1125        }
1126    }
1127  \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

13.20.1   Functions for before, after and first keys in enumext

__enumext_before_args_exec:
__enumext_before_keys_exec:
__enumext_after_stop_list:
__enumext_after_args_exec:

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

```
1128  \cs_new_protected:Nn \__enumext_before_args_exec:
1129    {
1130      \tl_use:c { l__enumext_before_starred_key_ \__enumext_level: _tl }
1131    }
```

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*⟩}}

```
1132  \cs_new_protected:Nn \__enumext_before_keys_exec:
1133    {
1134      \tl_use:c { l__enumext_before_no_starred_key_ \__enumext_level: _tl }
1135    }
```

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

```
1136 \cs_new_protected:Nn \__enumext_after_stop_list:
1137   {
1138     \tl_use:c { l__enumext_after_stop_list_ \__enumext_level: _tl }
1139   }
```

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`.

```
1140 \cs_new_protected:Nn \__enumext_after_args_exec:
1141   {
1142     \tl_use:c { l__enumext_after_list_args_ \__enumext_level: _tl }
1143   }
```

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

13.20.2   Functions for before, after and first keys in keyans

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

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

```
1144 \cs_new_protected:Nn \__enumext_before_args_exec_v:
1145   {
1146     \tl_use:N \l__enumext_before_starred_key_v_tl
1147   }
1148 \cs_new_protected:Nn \__enumext_before_keys_exec_v:
1149   {
1150     \tl_use:N \l__enumext_before_no_starred_key_v_tl
1151   }
1152 \cs_new_protected:Nn \__enumext_after_stop_list_v:
1153   {
1154     \tl_use:N \l__enumext_after_stop_list_v_tl
1155   }
1156 \cs_new_protected:Nn \__enumext_after_args_exec_v:
1157   {
1158     \tl_use:N \l__enumext_after_list_args_v_tl
1159   }
```

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

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

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

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

```
1160 \cs_new_protected:Nn \__enumext_before_args_exec_vii:
1161   {
1162     \tl_use:N \l__enumext_before_starred_key_vii_tl
1163   }
1164 \cs_new_protected:Nn \__enumext_before_args_exec_viii:
1165   {
1166     \tl_use:N \l__enumext_before_starred_key_viii_tl
1167   }
1168 \cs_new_protected:Nn \__enumext_before_keys_exec_vii:
1169   {
1170     \tl_use:N \l__enumext_before_no_starred_key_vii_tl
1171   }
1172 \cs_new_protected:Nn \__enumext_before_keys_exec_viii:
1173   {
1174     \tl_use:N \l__enumext_before_no_starred_key_viii_tl
1175   }
1176 \cs_new_protected:Nn \__enumext_after_stop_list_vii:
1177   {
1178     \tl_use:N \l__enumext_after_stop_list_vii_tl
1179   }
1180 \cs_new_protected:Nn \__enumext_after_stop_list_viii:
1181   {
1182     \tl_use:N \l__enumext_after_stop_list_viii_tl
1183   }
1184 \cs_new_protected:Nn \__enumext_after_args_exec_vii:
1185   {
1186     \tl_use:N \l__enumext_after_list_args_vii_tl
1187   }
1188 \cs_new_protected:Nn \__enumext_after_args_exec_viii:
1189   {
1190     \tl_use:N \l__enumext_after_list_args_viii_tl
1191   }
```

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

13.21   Setting keys for `multicols` and `minipage`

mini-env
mini-sep
columns-sep
columns

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

```
1192 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1193   {
1194     \keys_define:nn { enumext / #1 }
1195       {
1196         mini-env     .dim_set:c  = { l__enumext_minipage_right_#2_dim },
1197         mini-env     .value_required:n = true,
1198         mini-sep     .dim_set:c  = { l__enumext_minipage_hsep_#2_dim },
1199         mini-sep     .initial:n  = 0.3333em,
1200         mini-sep     .value_required:n = true,
1201         columns-sep .dim_set:c  = { l__enumext_columns_sep_#2_dim },
1202         columns-sep .value_required:n = true,
1203         columns      .int_set:c  = { l__enumext_columns_#2_int },
1204         columns      .initial:n  = 1,
1205         columns      .value_required:n = true,
1206       }
1207   }
1208 \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.

```
1209 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
1210   {
1211     \keys_define:nn { enumext / #1 }
1212       {
1213         mini-right  .tl_gset:c = { g__enumext_miniright_code_#2_tl },
1214         mini-right  .value_required:n = true,
1215         mini-right* .code:n    = {
1216                                   \bool_gset_true:c { g__enumext_minipage_center_#2_bool }
1217                                   \keys_set:nn { enumext / #1 } { mini-right = {##1} }
1218                                 },
1219         mini-right* .value_required:n = true,
1220       }
1221   }
1222 \clist_map_inline:nn { {enumext*}{vii}, {keyans*}{viii} } { \__enumext_tmp:nn #1 }
```

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

13.22   Adjustment of vertical spaces for `multicols`

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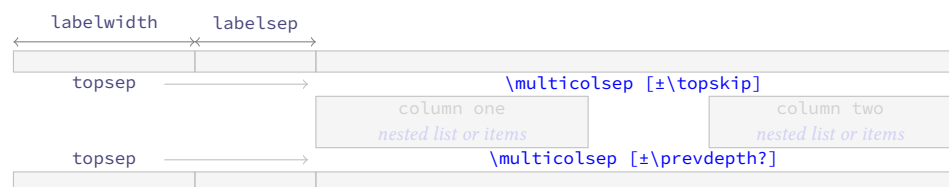
Figure 7: Representation of the vertical space in `multicols` for a nested level.

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.

13.22.1 Adjustment of vertical spaces for `multicols` in enumext

__enumext_multi_set_vskip:

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

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

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

```
1223 \cs_new_protected:Nn \__enumext_multi_set_vskip:
1224   {
1225     \skip_set:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1226       {
1227         \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1228       }
1229     \skip_set:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1230       {
1231         \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1232       }
1233     \__enumext_add_pre_parsep:
1234   }
```

(*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*.

```
1235 \cs_new_protected:Nn \__enumext_add_pre_parsep:
1236   {
1237     \int_case:nn { \l__enumext_level_int }
1238       {
1239         { 2 }{
1240             \skip_if_eq:nnF { \l__enumext_parsep_i_skip } { \c_zero_skip }
1241               {
1242                 \skip_add:Nn \l__enumext_multicols_above_ii_skip
1243                   {
1244                     \l__enumext_parsep_i_skip
1245                   }
1246               }
1247           }
1248         { 3 }{
1249             \skip_if_eq:nnF { \l__enumext_parsep_ii_skip } { \c_zero_skip }
1250               {
1251                 \skip_add:Nn \l__enumext_multicols_above_iii_skip
1252                   {
1253                     \l__enumext_parsep_ii_skip
1254                   }
1255               }
1256           }
1257         { 4 }{
1258             \skip_if_eq:nnF { \l__enumext_parsep_iii_skip } { \c_zero_skip }
1259               {
1260                 \skip_add:Nn \l__enumext_multicols_above_iv_skip
1261                   {
1262                     \l__enumext_parsep_iii_skip
1263                   }
1264               }
1265           }
1266       }
1267   }
```

(*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*⟩.

```
1268 \cs_new_protected:Nn \__enumext_multi_addvspace:
1269   {
1270     \__enumext_multi_set_vskip:
1271     \mode_if_vertical:T
1272       {
```

```
1273        \skip_add:cn { l__enumext_multicols_above_ \__enumext_level: _skip }
1274          {
1275            \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1276          }
1277        \skip_add:cn { l__enumext_multicols_below_ \__enumext_level: _skip }
1278          {
1279            \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1280          }
1281      }
1282    \par\nopagebreak
1283    \addvspace{ \skip_use:c { l__enumext_multicols_above_ \__enumext_level: _skip } }
1284  }
```

(*End of definition for* __enumext_multi_addvspace:.)

13.22.2 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.

```
1285 \cs_new_protected:Nn \__enumext_keyans_multi_set_vskip:
1286   {
1287     \skip_set:Nn \l__enumext_multicols_above_v_skip
1288       {
1289         \l__enumext_topsep_v_skip
1290       }
1291     \skip_set:Nn \l__enumext_multicols_below_v_skip
1292       {
1293         \l__enumext_topsep_v_skip
1294       }
1295   }
1296 \cs_new_protected:Nn \__enumext_keyans_multi_addvspace:
1297   {
1298     \__enumext_keyans_multi_set_vskip:
1299     \mode_if_vertical:T
1300       {
1301         \skip_add:Nn \l__enumext_multicols_above_v_skip
1302           {
1303             \skip_use:N \l__enumext_partopsep_v_skip
1304           }
1305         \skip_add:Nn \l__enumext_multicols_below_v_skip
1306           {
1307             \skip_use:N \l__enumext_partopsep_v_skip
1308           }
1309       }
1310     \par\nopagebreak
1311     \addvspace{ \l__enumext_multicols_above_v_skip }
1312   }
```

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

13.23 Adjustment of vertical spaces for `minipage`

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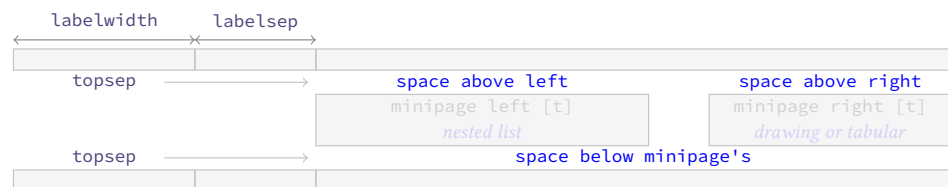
Figure 8: Representation of the minipage spacing adjustment for a nested level.

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.

13.23.1 Adjustment of vertical spaces for minipage in enumext

__enumext_minipage_set_skip:

__enumext_minipage_add_space:

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

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

```
1313 \cs_new_protected:Nn \__enumext_minipage_set_skip:
1314   {
1315     \skip_set:Nn \l__enumext_minipage_right_skip
1316       {
1317         \skip_use:c { l__enumext_topsep_ \__enumext_level: _skip }
1318       }
1319     \mode_if_vertical:T
1320       {
1321         \skip_add:Nn \l__enumext_minipage_right_skip
1322           {
1323             \skip_use:c { l__enumext_partopsep_ \__enumext_level: _skip }
1324           }
1325       }
1326     \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:.

```
1327     \skip_set_eq:cN
1328       { l__enumext_multicols_above_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1329     \skip_set_eq:cN
1330       { l__enumext_multicols_below_ \__enumext_level: _skip } \l__enumext_minipage_right_skip
1331     \__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.

```
1332     \int_compare:nNnT
1333       { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1334       {
1335         \skip_zero:N \topskip
1336         \skip_set_eq:Nc \multicolsep { l__enumext_multicols_above_ \__enumext_level: _skip }
1337       }
1338   }
```

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.*

```
1339 \cs_new_protected:Nn \__enumext_minipage_add_space:
1340   {
1341     \__enumext_minipage_set_skip:
1342     \__enumext_unskip_unkern:
1343     \mode_if_vertical:TF
1344       {
1345         \nopagebreak\nointerlineskip
1346       }
1347       {
1348         \par\nopagebreak\nointerlineskip
1349         \skip_zero:c { l__enumext_partopsep_ \__enumext_level: _skip }
1350       }
1351     \int_compare:nNnTF
1352       { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
1353       {
1354         \addvspace{ 0.445\box_ht:N \strutbox }
1355       }
1356       {
1357         \addvspace{ 0.250\box_ht:N \strutbox }
1358       }
1359   }
```

(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.

```
1360  \cs_new_protected:Nn \__enumext_pre_itemsep_skip:
1361    {
1362      \int_case:nn { \l__enumext_level_int }
1363        {
1364          { 2 }{
1365              \skip_if_eq:nnTF
1366                { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1367                {
1368                  \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1369                  \skip_set:Nn \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1370                }
1371                {
1372                  \dim_compare:nNnT
1373                    { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1374                    {
1375                      \skip_sub:Nn
1376                        \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1377                      \skip_sub:Nn
1378                        \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1379                      \skip_add:Nn
1380                        \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1381                      \skip_add:Nn
1382                        \l__enumext_multicols_below_ii_skip { 0.350\box_ht:N \strutbox }
1383                    }
1384                  \dim_compare:nNnT
1385                    { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1386                    {
1387                      \skip_set:Nn \l__enumext_minipage_temp_skip
1388                        {
1389                          \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1390                        }
1391                      \skip_sub:Nn
1392                        \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1393                      \skip_sub:Nn
1394                        \l__enumext_multicols_below_ii_skip { \l__enumext_itemsep_i_skip }
1395                      \skip_add:Nn
1396                        \l__enumext_minipage_after_skip
1397                        { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1398                      \skip_add:Nn
1399                        \l__enumext_multicols_below_ii_skip
1400                        { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1401                    }
1402                }
1403            }
1404          { 3 }{
1405              \skip_if_eq:nnTF
1406                { \l__enumext_itemsep_ii_skip } { \c_zero_skip }
1407                {
1408                  \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1409                  \skip_set:Nn \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1410                }
1411                {
1412                  \dim_compare:nNnT
1413                    { \l__enumext_itemsep_ii_skip } < { \l__enumext_minipage_after_skip }
1414                    {
1415                      \skip_sub:Nn
1416                        \l__enumext_minipage_after_skip { \l__enumext_itemsep_ii_skip }
1417                      \skip_sub:Nn
1418                        \l__enumext_multicols_below_iii_skip { \l__enumext_itemsep_ii_skip }
1419                      \skip_add:Nn
1420                        \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1421                      \skip_add:Nn
1422                        \l__enumext_multicols_below_iii_skip { 0.350\box_ht:N \strutbox }
1423                    }
1424                  \dim_compare:nNnT
1425                    { \l__enumext_itemsep_ii_skip } > { \l__enumext_minipage_after_skip }
```

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

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

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

`\__enumext_keyans_minipage_set_skip:`
`\__enumext_keyans_minipage_add_space:`
`\__enumext_keyans_pre_itemsep_skip:`

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

```
1485  \cs_new_protected:Nn \__enumext_keyans_minipage_set_skip:
1486    {
1487      \skip_zero:N \l__enumext_minipage_after_skip
1488      \skip_zero:N \l__enumext_minipage_left_skip
1489      \skip_zero:N \l__enumext_minipage_right_skip
```

```
1490        \skip_set:Nn \l__enumext_minipage_right_skip
1491          {
1492            \l__enumext_topsep_v_skip
1493          }
1494        \mode_if_vertical:T
1495          {
1496            \skip_add:Nn \l__enumext_minipage_right_skip
1497              {
1498                \l__enumext_partopsep_v_skip
1499              }
1500          }
1501        \skip_set_eq:NN \l__enumext_minipage_after_skip \l__enumext_minipage_right_skip
1502        \skip_set_eq:NN \l__enumext_multicols_above_v_skip \l__enumext_minipage_right_skip
1503        \skip_set_eq:NN \l__enumext_multicols_below_v_skip \l__enumext_minipage_right_skip
1504        \__enumext_keyans_pre_itemsep_skip:
1505        \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
1506          {
1507            \skip_zero:N \topskip
1508            \skip_set_eq:NN \multicolsep \l__enumext_minipage_right_skip
1509          }
1510      }
1511  \cs_new_protected:Nn \__enumext_keyans_minipage_add_space:
1512    {
1513      \__enumext_keyans_minipage_set_skip:
1514      \__enumext_unskip_unkern:
1515      \mode_if_vertical:TF
1516        {
1517          \nopagebreak\nointerlineskip
1518        }
1519        {
1520          \par\nopagebreak\nointerlineskip
1521          \skip_zero:N \l__enumext_partopsep_v_skip
1522        }
1523      \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
1524        {
1525          \addvspace{ 0.445\box_ht:N \strutbox }
1526        }
1527        {
1528          \addvspace{ 0.250\box_ht:N \strutbox }
1529        }
1530    }
1531  \cs_new_protected:Nn \__enumext_keyans_pre_itemsep_skip:
1532    {
1533      \skip_if_eq:nnTF
1534        { \l__enumext_itemsep_i_skip } { \l__enumext_minipage_after_skip }
1535        {
1536          \skip_set:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1537          \skip_set:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1538        }
1539        {
1540          \dim_compare:nNnT
1541            { \l__enumext_itemsep_i_skip } < { \l__enumext_minipage_after_skip }
1542            {
1543              \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1544              \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1545              \skip_add:Nn \l__enumext_minipage_after_skip { 0.150\box_ht:N \strutbox }
1546              \skip_add:Nn \l__enumext_multicols_below_v_skip { 0.350\box_ht:N \strutbox }
1547            }
1548          \dim_compare:nNnT
1549            { \l__enumext_itemsep_i_skip } > { \l__enumext_minipage_after_skip }
1550            {
1551              \skip_set:Nn \l__enumext_minipage_temp_skip
1552                {
1553                  \l__enumext_itemsep_i_skip - \l__enumext_minipage_after_skip
1554                }
1555              \skip_sub:Nn \l__enumext_minipage_after_skip { \l__enumext_itemsep_i_skip }
1556              \skip_sub:Nn \l__enumext_multicols_below_v_skip { \l__enumext_itemsep_i_skip }
1557              \skip_add:Nn \l__enumext_minipage_after_skip
1558                { 0.150\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
1559              \skip_add:Nn \l__enumext_multicols_below_v_skip
1560                { 0.350\box_ht:N \strutbox + \l__enumext_minipage_temp_skip }
```

```
1561                    }
1562            }
1563        }
```

(*End of definition for* `\__enumext_keyans_minipage_set_skip:`, `\__enumext_keyans_minipage_add_space:`, *and* `\__enumext_keyans_pre_itemsep_skip:`.)

13.23.3 Adjustment of vertical spaces for `minipage` in `enumext*` and `keyans*`

`\__enumext_mini_set_vskip_vii:`
`\__enumext_mini_set_vskip_viii:`

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

```
1564 \cs_new_protected:Nn \__enumext_mini_set_vskip_vii:
1565    {
1566        \skip_zero_new:N \l__enumext_minipage_left_skip
1567        \skip_gzero_new:N \g__enumext_minipage_right_skip
1568        \skip_gzero_new:N \g__enumext_minipage_after_skip
1569        \skip_if_eq:nnTF { \l__enumext_topsep_vii_skip } { \c_zero_skip }
1570          {
1571            \skip_set:Nn \l__enumext_minipage_left_skip { 0.5\box_dp:N \strutbox }
1572            \skip_gset:Nn \g__enumext_minipage_right_skip { 0.325\box_dp:N \strutbox }
1573          }
1574          {
1575            \skip_set:Nn \l__enumext_minipage_left_skip { 0.5875\box_dp:N \strutbox }
1576            \skip_gset:Nn \g__enumext_minipage_right_skip
1577              {
1578                \l__enumext_topsep_vii_skip
1579              }
1580            \skip_gset:Nn \g__enumext_minipage_after_skip
1581              {
1582                0.325\box_dp:N \strutbox + \l__enumext_topsep_vii_skip
1583              }
1584          }
1585    }
1586 \cs_new_protected:Nn \__enumext_mini_set_vskip_viii:
1587    {
1588        \skip_zero_new:N \l__enumext_minipage_after_skip
1589        \skip_zero_new:N \l__enumext_minipage_left_skip
1590        \skip_zero_new:N \l__enumext_minipage_right_skip
1591        \skip_if_eq:nnTF { \l__enumext_topsep_viii_skip } { \c_zero_skip }
1592          {
1593            \skip_set:Nn \l__enumext_minipage_left_skip
1594              {
1595                0.5\box_dp:N \strutbox
1596              }
1597            \skip_set:Nn \l__enumext_minipage_right_skip
1598              {
1599                \l__enumext_partopsep_viii_skip
1600              }
1601            \skip_set:Nn \l__enumext_minipage_after_skip
1602              {
1603                1.6\box_dp:N \strutbox
1604              }
1605          }
1606          {
1607            \skip_set:Nn \l__enumext_minipage_left_skip
1608              {
1609                0.5875\box_dp:N \strutbox
1610              }
1611            \skip_set:Nn \l__enumext_minipage_right_skip
1612              {
1613                \l__enumext_topsep_viii_skip
1614              }
1615            \skip_set:Nn \l__enumext_minipage_after_skip
1616              {
1617                0.325\box_dp:N \strutbox + \l__enumext_topsep_viii_skip
1618              }
1619          }
1620    }
```

(*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.

```
1621 \cs_new_protected:Nn \__enumext_mini_addvspace_vii:
1622   {
1623     \__enumext_mini_set_vskip_vii:
1624     \par\nopagebreak
1625     \addvspace { \l__enumext_minipage_left_skip }
1626   }
1627 \cs_new_protected:Nn \__enumext_mini_addvspace_viii:
1628   {
1629     \__enumext_mini_set_vskip_viii:
1630     \par\nopagebreak
1631     \addvspace { \l__enumext_minipage_left_skip }
1632   }
```

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

13.23.4 The command \miniright

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

\miniright

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

```
1633 \NewDocumentCommand \miniright { s }
1634   {
1635     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
1636       {
1637         \msg_error:nnn { enumext } { wrong-miniright-place }
1638       }
1639     % outside
1640     \bool_lazy_and:nnT
1641       { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
1642       { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
1643       {
1644         \msg_error:nnn { enumext } { wrong-miniright-place }
1645       }
1646     % starred env
1647     \bool_lazy_and:nnT
1648       { \bool_if_p:N \g__enumext_starred_bool }
1649       { \bool_not_p:n { \l__enumext_standar_bool } }
1650       {
1651         \msg_error:nnn { enumext } { wrong-miniright-starred }
1652       }
1653     % exec
1654     \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
1655       {
1656         \__enumext_keyans_mini_right_cmd:n {#1}
1657       }
1658       { \__enumext_mini_right_cmd:n {#1} }
1659   }
```

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

__enumext_mini_right_cmd:n

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

```
1660 \cs_new_protected:Npn \__enumext_mini_right_cmd:n #1
1661   {
1662     \dim_compare:nNnTF
1663       { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
```

```
1664    {
1665        \__enumext_multicols_stop:
1666        \int_compare:nNnT
1667          { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } = { 1 }
1668          {
1669            \par\addvspace{ \l__enumext_minipage_after_skip }
1670          }
1671        \end__enumext_mini_page
1672        \hfill
1673        \__enumext_mini_page{ \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } }
1674          \par\nointerlineskip
1675          \addvspace { \l__enumext_minipage_right_skip }
1676          \bool_if:nF {#1}
1677            {
1678              \centering
1679            }
1680          \int_gzero:N \g__enumext_minipage_stat_int
1681      }
1682      { \msg_error:nnn { enumext } { wrong-miniright-use } }
1683    % paranoia
1684    \RenewDocumentCommand \miniright { s }
1685      {
1686        \msg_error:nn { enumext } { many-miniright-used }
1687      }
1688  }
```

(*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.

```
1689  \cs_new_protected:Npn \__enumext_keyans_mini_right_cmd:n #1
1690    {
1691      \dim_compare:nNnTF { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
1692        {
1693          \__enumext_keyans_multicols_stop:
1694          \int_compare:nNnT { \l__enumext_columns_v_int } = { 1 }
1695            {
1696              \par\addvspace{ \l__enumext_minipage_after_skip }
1697            }
1698          \end__enumext_mini_page
1699          \hfill
1700          \__enumext_mini_page{ \l__enumext_minipage_right_v_dim }
1701            \par\nointerlineskip
1702            \addvspace { \l__enumext_minipage_right_skip }
1703            \bool_if:nF {#1}
1704              {
1705                \centering
1706              }
1707            \int_gzero:N \g__enumext_minipage_stat_int
1708        }
1709        { \msg_error:nnn { enumext } { wrong-miniright-use } }
1710      % paranoia
1711      \RenewDocumentCommand \miniright { s }
1712        {
1713          \msg_error:nn { enumext } { many-miniright-used }
1714        }
1715  }
```

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

13.24   Setting above and below keys

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

above
above*
below
below*

Define above, above*, below and below* keys for enumext and keyans environments.

```
1716  \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
```

```
1717   {
1718     \keys_define:nn { enumext / #1 }
1719       {
1720         above  .skip_set:c = { l__enumext_vspace_above_#2_skip },
1721         above  .value_required:n = true,
1722         above* .code:n      = \bool_set_true:c { l__enumext_vspace_a_star_#2_bool }
1723                               \keys_set:nn { enumext / #1 } { above = {##1} },
1724         above* .value_required:n = true,
1725         below  .skip_set:c = { l__enumext_vspace_below_#2_skip },
1726         below  .value_required:n = true,
1727         below* .code:n      = \bool_set_true:c { l__enumext_vspace_b_star_#2_bool }
1728                               \keys_set:nn { enumext / #1 } { below = {##1} },
1729         below* .value_required:n = true,
1730       }
1731   }
1732 \clist_map_inline:Nn \c__enumext_all_envs_clist { \__enumext_tmp:nn #1 }
```

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

13.24.1   Functions for above and below keys in enumext

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

```
1733 \cs_new_protected:Nn \__enumext_vspace_above:
1734   {
1735     \skip_if_eq:nnF
1736       { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } } { \c_zero_skip }
1737       {
1738         \bool_if:cTF { l__enumext_vspace_a_star_ \__enumext_level: _bool }
1739           {
1740             \vspace*{ \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1741           }
1742           {
1743             \vspace { \skip_use:c { l__enumext_vspace_above_ \__enumext_level: _skip } }
1744           }
1745       }
1746   }
```

(*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.

```
1747 \cs_new_protected:Nn \__enumext_vspace_below:
1748   {
1749     \skip_if_eq:nnF
1750       { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } } { \c_zero_skip }
1751       {
1752         \bool_if:cTF { l__enumext_vspace_b_star_ \__enumext_level: _bool }
1753           {
1754             \vspace*{ \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1755           }
1756           {
1757             \vspace { \skip_use:c { l__enumext_vspace_below_ \__enumext_level: _skip } }
1758           }
1759       }
1760   }
```

(*End of definition for __enumext_vspace_below:.*)

13.24.2   Functions for above and below keys in keyans

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

```
1761 \cs_new_protected:Nn \__enumext_vspace_above_v:
1762   {
1763     \skip_if_eq:nnF { \l__enumext_vspace_above_v_skip } { \c_zero_skip }
1764       {
1765         \bool_if:NTF \l__enumext_vspace_a_star_v_bool
1766           {
1767             \vspace*{ \l__enumext_vspace_above_v_skip }
1768           }
1769           { \vspace { \l__enumext_vspace_above_v_skip } }
1770       }
1771   }
```

(*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.

```
1772 \cs_new_protected:Nn \__enumext_vspace_below_v:
1773   {
1774     \skip_if_eq:nnF { \l__enumext_vspace_below_v_skip } { \c_zero_skip }
1775       {
1776         \bool_if:NTF \l__enumext_vspace_b_star_v_bool
1777           {
1778             \vspace*{ \l__enumext_vspace_below_v_skip }
1779           }
1780           { \vspace { \l__enumext_vspace_below_v_skip } }
1781       }
1782   }
```

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

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

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

```
1783 \cs_new_protected:Nn \__enumext_vspace_above_vii:
1784   {
1785     \skip_if_eq:nnF { \l__enumext_vspace_above_vii_skip } { \c_zero_skip }
1786       {
1787         \bool_if:NTF \l__enumext_vspace_a_star_vii_bool
1788           {
1789             \vspace*{ \l__enumext_vspace_above_vii_skip }
1790           }
1791           { \vspace { \l__enumext_vspace_above_vii_skip } }
1792       }
1793   }
1794 \cs_new_protected:Nn \__enumext_vspace_above_viii:
1795   {
1796     \skip_if_eq:nnF { \l__enumext_vspace_above_viii_skip } { \c_zero_skip }
1797       {
1798         \bool_if:NTF \l__enumext_vspace_a_star_viii_bool
1799           {
1800             \vspace*{ \l__enumext_vspace_above_viii_skip }
1801           }
1802           { \vspace { \l__enumext_vspace_above_viii_skip } }
1803       }
1804   }
```

(*End of definition for* __enumext_vspace_above_vii: *and* __enumext_vspace_above_viii:*.*)

__enumext_vspace_below_vii: The functions __enumext_vspace_below_vii: and __enumext_vspace_below_viii: apply the *verti-*
__enumext_vspace_below_viii: *cal space below* the enumext* and keyans* environments set by the below* and below keys.

```
1805 \cs_new_protected:Nn \__enumext_vspace_below_vii:
1806   {
1807     \skip_if_eq:nnF { \l__enumext_vspace_below_vii_skip } { \c_zero_skip }
1808       {
1809         \bool_if:NTF \l__enumext_vspace_b_star_vii_bool
1810           {
1811             \vspace*{ \l__enumext_vspace_below_vii_skip }
1812           }
1813           { \vspace { \l__enumext_vspace_below_vii_skip } }
1814       }
1815   }
1816 \cs_new_protected:Nn \__enumext_vspace_below_viii:
1817   {
1818     \skip_if_eq:nnF { \l__enumext_vspace_below_viii_skip } { \c_zero_skip }
1819       {
1820         \bool_if:NTF \l__enumext_vspace_b_star_viii_bool
1821           {
1822             \vspace*{ \l__enumext_vspace_below_viii_skip }
1823           }
1824           { \vspace { \l__enumext_vspace_below_viii_skip } }
1825       }
1826   }
```

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

13.25 Setting `series`, `resume` and `resume*` keys

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

series
resume
resume*

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

```
1827 \cs_set_protected:Npn \__enumext_tmp:n #1
1828   {
1829     \keys_define:nn { enumext / #1 }
1830       {
1831         series  .str_set:N = \l__enumext_series_str,
1832         series  .value_required:n = true,
1833         resume  .code:n = \__enumext_resume_series:n {##1},
1834         resume* .code:n = \__enumext_resume_starred:,
1835         resume* .value_forbidden:n = true,
1836       }
1837   }
1838 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

13.25.1 Internal functions for series key

__enumext_filter_series:n
__enumext_filter_series_key:n
__enumext_filter_series_pair:nn

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

```
1839 \cs_new:Npn \__enumext_filter_series:n #1
1840   {
1841     \use:e
1842       {
1843         \keyval_parse:NNn
1844           \__enumext_filter_series_key:n
1845           \__enumext_filter_series_pair:nn {#1}
1846       }
1847   }
```

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.

```
1848 \cs_new:Npn \__enumext_filter_series_key:n #1
1849   {
1850     \str_case:nnF {#1}
1851       {
1852         { resume } {} { resume* } {} { base-fix } {}
1853       }
1854       { , { \exp_not:n {#1} } }
1855   }
```

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.

```
1856 \cs_new:Npn \__enumext_filter_series_pair:nn #1#2
1857   {
1858     \str_case:nnF {#1}
1859       {
1860         { series } {} { resume } {} { start } {}
1861         { start* } {}  { save-ans } {} { save-key } {}
1862       }
1863       { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
1864   }
```

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

__enumext_parse_series:n
__enumext_resume_last:n

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

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

```
1865  \cs_new_protected:Npn \__enumext_parse_series:n #1
1866    {
1867      \str_if_empty:NTF \l__enumext_series_str
1868        {
1869          \bool_if:NF \l__enumext_resume_active_bool
1870            {
1871              \__enumext_resume_last:n {#1}
1872            }
1873        }
1874        {
1875          \tl_gclear_new:c { g__enumext_series_ \l__enumext_series_str _tl }
1876          \tl_gset:ce { g__enumext_series_ \l__enumext_series_str _tl }
1877            { \__enumext_filter_series:n {#1} }
1878          \int_if_exist:cF { g__enumext_series_ \l__enumext_series_str _int }
1879            {
1880              \int_new:c { g__enumext_series_ \l__enumext_series_str _int }
1881            }
1882        }
1883    }
```

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.

```
1884  \cs_new_protected:Npn \__enumext_resume_last:n #1
1885    {
1886      \bool_if:NT \l__enumext_standar_first_bool
1887        {
1888          \tl_gclear:N \g__enumext_standar_series_tl
1889          \tl_gset:Ne \g__enumext_standar_series_tl { \__enumext_filter_series:n {#1} }
1890        }
1891      \bool_if:NT \l__enumext_starred_first_bool
1892        {
1893          \tl_gclear:N \g__enumext_starred_series_tl
1894          \tl_gset:Ne \g__enumext_starred_series_tl { \__enumext_filter_series:n {#1} }
1895        }
1896    }
```

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

13.25.2   Internal function to save counter value

__enumext_resume_save_counter:

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

🧨 The variables \l__enumext_series_str and \l__enumext__resume_name_tl contain the same {⟨series name⟩} but are executed at different moments, the integer variable with \l__enumext_series_str sets the value when execute series={⟨series name⟩} and the integer variable with \l__enumext__resume_name_tl sets the subsequent values when use resume={⟨series name⟩}. This function is passed to the enumext environment definition (§13.39) and the enumext* environment definition (§13.44).

```
1897  \cs_new_protected:Nn \__enumext_resume_save_counter:
1898    {
1899      \bool_if:NT \g__enumext_standar_bool
1900        {
1901          \tl_if_empty:NF \l__enumext_series_str
1902            {
1903              \int_gset_eq:cN
1904                { g__enumext_series_ \l__enumext_series_str _int } \value{enumXi}
1905            }
1906          \tl_if_empty:NTF \l__enumext_resume_name_tl
1907            {
1908              \str_if_empty:NT \l__enumext_series_str
1909                {
1910                  \int_gset_eq:NN \g__enumext_resume_int \value{enumXi}
1911                }
1912            }
1913            {
1914              \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1915                {
1916                  \int_gset_eq:cN
```

```
1917                    { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXi}
1918                }
1919            }
1920        \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1921            {
1922                \int_gset_eq:cN
1923                    { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXi}
1924            }
1925        }
1926    \bool_if:NT \g__enumext_starred_bool
1927        {
1928            \tl_if_empty:NF \l__enumext_series_str
1929                {
1930                    \int_gset_eq:cN
1931                        { g__enumext_series_ \l__enumext_series_str _int } \value{enumXvii}
1932                }
1933            \tl_if_empty:NTF \l__enumext_resume_name_tl
1934                {
1935                    \str_if_empty:NT \l__enumext_series_str
1936                        {
1937                            \int_gset_eq:NN \g__enumext_resume_vii_int \value{enumXvii}
1938                        }
1939                }
1940                {
1941                    \int_if_exist:cT { g__enumext_series_ \l__enumext_resume_name_tl _int }
1942                        {
1943                            \int_gset_eq:cN
1944                                { g__enumext_series_ \l__enumext_resume_name_tl _int } \value{enumXvii}
1945                        }
1946                }
1947            \int_if_exist:cT { g__enumext_resume_ \l__enumext_store_name_tl _int }
1948                {
1949                    \int_gset_eq:cN
1950                        { g__enumext_resume_ \l__enumext_store_name_tl _int } \value{enumXvii}
1951                }
1952        }
1953    }
```

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

13.25.3   Internal functions for resume key

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

```
1954  \cs_new_protected:Npn \__enumext_resume_series:n #1
1955    {
1956      \tl_if_empty:nTF {#1}
1957        {
1958          \__enumext_resume_counter:n { }
1959        }
1960        {
1961          \tl_if_exist:cTF { g__enumext_series_ \tl_to_str:n {#1} _tl }
1962            {
1963              \__enumext_resume_counter:n {#1}
1964              \bool_if:NT \g__enumext_standar_bool
1965                {
1966                  \keys_set:nv { enumext / level-1 }
1967                    { g__enumext_series_ \tl_to_str:n {#1} _tl }
1968                }
1969              \bool_if:NT \g__enumext_starred_bool
1970                {
1971                  \keys_set:nv { enumext / enumext* }
1972                    { g__enumext_series_ \tl_to_str:n {#1} _tl }
1973                }
1974            }
1975            {
1976              \bool_if:NT \g__enumext_standar_bool
```

```
1977                    {
1978                        \msg_error:nnn { enumext } { unknown-series } {#1}
1979                    }
1980                \bool_if:NT \g__enumext_starred_bool
1981                    {
1982                        \msg_error:nnn { enumext } { unknown-series } {#1}
1983                    }
1984            }
1985        }
1986    }
```

(*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.

```
1987 \cs_new_protected:Npn \__enumext_resume_counter:n #1
1988    {
1989        \bool_set_true:N \l__enumext_resume_active_bool
1990        \tl_set:Nn \l__enumext_resume_name_tl {#1}
1991        \tl_if_empty:NTF \l__enumext_resume_name_tl
1992            {
1993                \__enumext_resume_counter:
1994            }
1995            {
1996                \__enumext_resume_counter_series:
1997            }
1998        \__enumext_resume_counter_save_ans:
1999    }
```

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.

```
2000 \cs_new_protected:Nn \__enumext_resume_counter:
2001    {
2002        \bool_if:NT \g__enumext_standar_bool
2003            {
2004                \int_gincr:N \g__enumext_resume_int
2005                \int_set_eq:NN \l__enumext_start_i_int \g__enumext_resume_int
2006            }
2007        \bool_if:NT \g__enumext_starred_bool
2008            {
2009                \int_gincr:N \g__enumext_resume_vii_int
2010                \int_set_eq:NN \l__enumext_start_vii_int \g__enumext_resume_vii_int
2011            }
2012    }
```

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.

```
2013 \cs_new_protected:Nn \__enumext_resume_counter_series:
2014    {
2015        \bool_if:NT \g__enumext_standar_bool
2016            {
2017                \int_set:Nn \l__enumext_start_i_int
2018                    {
2019                        \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
2020                    }
2021            }
2022        \bool_if:NT \g__enumext_starred_bool
2023            {
2024                \int_set:Nn \l__enumext_start_vii_int
2025                    {
2026                        \int_use:c { g__enumext_series_ \l__enumext_resume_name_tl _int } + 1
2027                    }
2028            }
2029    }
```

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.

```
2030  \cs_new_protected:Nn \__enumext_resume_counter_save_ans:
2031    {
2032      \bool_lazy_and:nnT
2033        { \bool_if_p:N \l__enumext_standar_first_bool }
2034        { \bool_if_p:N \l__enumext_store_active_bool }
2035        {
2036          \int_set:Nn \l__enumext_start_i_int
2037            {
2038              \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
2039            }
2040        }
2041      \bool_lazy_and:nnT
2042        { \bool_if_p:N \l__enumext_starred_first_bool }
2043        { \bool_if_p:N \l__enumext_store_active_bool }
2044        {
2045          \int_set:Nn \l__enumext_start_vii_int
2046            {
2047              \int_use:c { g__enumext_resume_ \l__enumext_store_name_tl _int } + 1
2048            }
2049        }
2050    }
```

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

13.25.4 Internal function for resume* key

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

```
2051  \cs_new_protected:Nn \__enumext_resume_starred:
2052    {
2053      \bool_if:NT \g__enumext_standar_bool
2054        {
2055          \tl_if_empty:NF \g__enumext_standar_series_tl
2056            {
2057              \__enumext_resume_counter:n { }
2058              \keys_set:nV { enumext / level-1 } \g__enumext_standar_series_tl
2059            }
2060        }
2061      \bool_if:NT \g__enumext_starred_bool
2062        {
2063          \tl_if_empty:NF \g__enumext_starred_series_tl
2064            {
2065              \__enumext_resume_counter:n { }
2066              \keys_set:nV { enumext / enumext* } \g__enumext_starred_series_tl
2067            }
2068        }
2069    }
```

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

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

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

13.26.1 Setting save-ans key

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

```
2070  \cs_set_protected:Npn \__enumext_tmp:n #1
2071    {
2072      \keys_define:nn { enumext / #1 }
2073        {
2074          save-ans .code:n = \__enumext_storing_set:n {##1},
2075          save-ans .value_required:n = true,
2076        }
2077    }
2078  \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

13.26.2 Internal functions for save-ans key

__enumext_start_save_ans_msg:
__enumext_stop_save_ans_msg:

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

```
2079  \cs_new_protected:Nn \__enumext_start_save_ans_msg:
2080    {
2081      \msg_term:nnVV { enumext } { save-ans-log }
2082        \g__enumext_envir_name_tl \l__enumext_store_name_tl
2083    }
2084  \cs_new_protected:Nn \__enumext_stop_save_ans_msg:
2085    {
2086      \msg_term:nnVV { enumext } { save-ans-log-hook }
2087        \g__enumext_envir_name_tl \g__enumext_store_name_tl
2088    }
```

(*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.

```
2089  \cs_new_protected:Npn \__enumext_storing_set:n #1
2090    {
2091      \tl_set:Ne \l__enumext_store_name_tl {#1}
2092      \tl_if_empty:NTF \l__enumext_store_name_tl
2093        {
2094          \bool_lazy_or:nnT
2095            { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
2096            {
2097              \msg_error:nnV { enumext } { save-ans-empty } \g__enumext_envir_name_tl
2098            }
2099        }
2100        {
2101          \bool_lazy_or:nnT
2102            { \l__enumext_standar_first_bool } { \l__enumext_starred_first_bool }
2103            {
2104              \__enumext_start_save_ans_msg:
2105              \__enumext_storing_exec:
2106            }
2107        }
2108    }
```

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

```
2109  \cs_new_protected:Nn \__enumext_storing_exec:
2110    {
2111      \bool_set_true:N \l__enumext_store_active_bool
2112      \bool_set_true:N \l__enumext_check_answers_bool
2113      \tl_gset:NV \g__enumext_store_name_tl \l__enumext_store_name_tl
2114      \__enumext_anskey_env_make:V \l__enumext_store_name_tl
```

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

```
2115      \prop_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _prop }
2116        {
2117          \msg_log:nnV { enumext } { store-prop } \l__enumext_store_name_tl
2118          \prop_new:c { g__enumext_ \l__enumext_store_name_tl _prop }
2119        }
2120      \seq_if_exist:cF { g__enumext_ \l__enumext_store_name_tl _seq }
2121        {
2122          \msg_log:nnV { enumext } { store-seq } \l__enumext_store_name_tl
2123          \seq_new:c { g__enumext_ \l__enumext_store_name_tl _seq }
```

```
2124          }
2125      \int_if_exist:cF { g__enumext_resume_ \l__enumext_store_name_tl _int }
2126        {
2127          \msg_log:nnV { enumext } { store-int } \l__enumext_store_name_tl
2128          \int_new:c { g__enumext_resume_ \l__enumext_store_name_tl _int }
2129        }
2130    }
```

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

13.26.3   The check answer mechanism

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

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

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

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

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

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

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

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

13.26.4   Setting check-ans and no-store keys

check-ans
no-store

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

```
2131 \cs_set_protected:Npn \__enumext_tmp:n #1
2132   {
2133     \keys_define:nn { enumext / #1 }
2134       {
2135         check-ans .bool_set:N = \l__enumext_check_ans_key_bool,
2136         check-ans .initial:n  = false,
2137         check-ans .value_required:n = true,
2138         no-store  .code:n = {
2139                               \bool_set_false:N \l__enumext_check_answers_bool
2140                               \bool_set_false:N \l__enumext_check_ans_key_bool
2141                             },
2142         no-store  .value_forbidden:n = true,
2143       }
2144   }
2145 \clist_map_inline:nn
2146   {
2147     level-1, level-2, level-3, level-4, enumext*
2148   }
2149   { \__enumext_tmp:n {#1} }
```

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

13.26.5   Set-up check answer mechanism

__enumext_check_ans_active: The function `\__enumext_check_ans_active:` will first check the state of the variable `\l__enumext_-`
__enumext_check_ans_level: `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.

```
2150  \cs_new_protected:Nn \__enumext_check_ans_active:
2151    {
2152      \tl_if_empty:NF \l__enumext_store_name_tl
2153        {
2154          \bool_if:NT \l__enumext_check_answers_bool
2155            {
2156              \__enumext_check_ans_level:
2157            }
2158        }
2159    }
```

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"* .

```
2160  \cs_new_protected:Nn \__enumext_check_ans_level:
2161    {
2162      \int_case:nn { \l__enumext_level_int }
2163        {
2164          { 1 }{
2165                \bool_lazy_all:nT
2166                  {
2167                    { \bool_if_p:N \g__enumext_starred_bool }
2168                    { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
2169                  }
2170                  {
2171                    \int_gdecr:N \g__enumext_item_number_int
2172                    \bool_set_false:N \l__enumext_item_number_bool
2173                  }
2174            }
2175          { 2 }{
2176                \int_gdecr:N \g__enumext_item_number_int
2177                \bool_set_false:N \l__enumext_item_number_bool
2178            }
2179          { 3 }{
2180                \int_gdecr:N \g__enumext_item_number_int
2181                \bool_set_false:N \l__enumext_item_number_bool
2182            }
2183          { 4 }{
2184                \int_gdecr:N \g__enumext_item_number_int
2185                \bool_set_false:N \l__enumext_item_number_bool
2186            }
2187        }
```

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.

```
2188      \int_case:nn { \l__enumext_level_h_int }
2189        {
2190          { 1 }{
2191                \bool_lazy_all:nT
2192                  {
2193                    { \bool_if_p:N \g__enumext_standar_bool }
2194                    { \int_compare_p:nNn { \l__enumext_level_int } = { 1 } }
2195                  }
2196                  {
2197                    \int_gdecr:N \g__enumext_item_number_int
2198                    \bool_set_false:N \l__enumext_item_number_bool
2199                  }
2200            }
2201        }
2202    }
```

(*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.

```
2203 \cs_new_protected:Nn \__enumext_check_ans_key_hook:
2204   {
2205     \bool_lazy_and:nnT
2206       { \bool_if_p:N \l__enumext_check_ans_key_bool }
2207       { \bool_if_p:N \g__enumext_standar_bool }
2208       {
2209         \bool_gset_true:N \g__enumext_check_ans_key_bool
2210       }
2211     \bool_lazy_and:nnT
2212       { \bool_if_p:N \l__enumext_check_ans_key_bool }
2213       { \bool_if_p:N \g__enumext_starred_bool }
2214       {
2215         \bool_gset_true:N \g__enumext_check_ans_key_bool
2216       }
2217   }
```

(*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:.

```
2218 \cs_new_protected:Nn \__enumext_item_answer_diff:
2219   {
2220     \int_gset:Nn \g__enumext_item_answer_diff_int
2221       {
2222         \int_sign:n { \g__enumext_item_number_int - \g__enumext_item_anskey_int }
2223       }
2224   }
```

(*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:.

```
2225 \cs_new_protected:Nn \__enumext_check_ans_show:
2226   {
2227     \int_case:nn { \g__enumext_item_answer_diff_int }
2228       {
2229         { -1 }{ \__enumext_check_ans_msg_less:     }
2230         {  0 }{ \__enumext_check_ans_msg_same_ok: }
2231         {  1 }{ \__enumext_check_ans_msg_greater: }
2232       }
2233   }
2234 \cs_new_protected:Nn \__enumext_check_ans_msg_less:
2235   {
2236     \msg_warning:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2237       { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2238   }
2239 \cs_new_protected:Nn \__enumext_check_ans_msg_same_ok:
2240   {
2241     \msg_term:nneee { enumext } { items-same-answer } { \g__enumext_store_name_tl }
2242       { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2243   }
2244 \cs_new_protected:Nn \__enumext_check_ans_msg_greater:
2245   {
2246     \msg_warning:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2247       { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2248   }
```

(*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:.

```
2249  \cs_new_protected:Nn \__enumext_check_ans_log:
2250    {
2251      \int_case:nn { \g__enumext_item_answer_diff_int }
2252        {
2253          { -1 }{ \__enumext_check_ans_log_msg_less:     }
2254          {  0 }{ \__enumext_check_ans_log_msg_same_ok: }
2255          {  1 }{ \__enumext_check_ans_log_msg_greater: }
2256        }
2257    }
2258  \cs_new_protected:Nn \__enumext_check_ans_log_msg_less:
2259    {
2260      \msg_log:nneee { enumext } { item-less-answer } { \g__enumext_store_name_tl }
2261        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2262    }
2263  \cs_new_protected:Nn \__enumext_check_ans_log_msg_same_ok:
2264    {
2265      \msg_log:nneee { enumext } { items-same-answer }  { \g__enumext_store_name_tl }
2266        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2267    }
2268  \cs_new_protected:Nn \__enumext_check_ans_log_msg_greater:
2269    {
2270      \msg_log:nneee { enumext } { item-greater-answer } { \g__enumext_store_name_tl }
2271        { \g__enumext_envir_name_tl } { \g__enumext_start_line_tl }
2272    }
```

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

13.26.6   Check for \item* and \anspic* commands

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

```
2273  \cs_new_protected:Npn \__enumext_check_starred_cmd:n #1
2274    {
2275      \int_compare:nNnT
2276        { \g__enumext_check_starred_cmd_int } = { 0 }
2277        {
2278          \msg_warning:nnnV
2279            { enumext } { missing-starred }{ #1 } \l__enumext_check_start_line_env_tl
2280        }
2281      \int_compare:nNnT
2282        { \g__enumext_check_starred_cmd_int } > { 1 }
2283        {
2284          \msg_warning:nnnV
2285            { enumext } { many-starred }{ #1 } \l__enumext_check_start_line_env_tl
2286        }
2287      \int_gzero:N \g__enumext_check_starred_cmd_int
2288      \tl_clear:N \l__enumext_check_start_line_env_tl
2289    }
```

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

13.27   Keys and functions associated with storage

wrap-ans   We add the keys wrap-ans, wrap-opt, wrap-key,save-sep, mark-ans, mark-pos, mark-sep, show-ans,
wrap-opt   show-pos, mark-ref and save-ref related to the *"storage system"* and internal mechanism of *"label and ref"*
wrap-key   only at the *first level* of enumext and enumext*.
save-sep
mark-ans   ```
mark-pos   2290 \cs_set_protected:Npn __enumext_tmp:n #1
mark-sep   2291   {
show-ans   2292   \keys_define:nn { enumext / #1 }
mark-ref   2293   {
save-ref   2294   wrap-ans   .cs_set_protected:Np = __enumext_anskey_wrapper:n ##1,
           2295   wrap-ans   .initial:n =
           2296   {
           2297   \fbox{\parbox[t]{\dimeval{\itemwidth -2\fboxsep -2\fboxrule}}{##1}}
           2298   },
           2299   wrap-ans   .value_required:n = true,
           2300   wrap-opt   .cs_set_protected:Np = __enumext_keyans_wrapper_opt:n ##1,
           2301   wrap-opt   .initial:n = [{##1}],
           2302   wrap-opt   .value_required:n = true,
           2303   wrap-key   .cs_set_protected:Np = __enumext_keyans_wrapper_item:n ##1,
           2304   wrap-key   .value_required:n = true,
```

```
2305         save-sep   .tl_set:N   = \l__enumext_store_keyans_item_opt_sep_tl,
2306         save-sep   .initial:n = {, ~ },
2307         save-sep   .value_required:n = true,
2308         mark-ans   .tl_set:N   = \l__enumext_mark_answer_sym_tl,
2309         mark-ans   .initial:n = \textasteriskcentered,
2310         mark-ans   .value_required:n = true,
2311         mark-pos   .choice:,
2312         mark-pos / left    .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2313         mark-pos / right   .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2314         mark-pos / center  .code:n = \str_set:Nn \l__enumext_mark_position_str { c },
2315         mark-pos / unknown .code:n =
2316                         \msg_error:nneee { enumext } { unknown-choice }
2317                           { mark-pos } { left, ~ right, ~ center } { \exp_not:n {##1} },
2318         mark-pos   .initial:n = right,
2319         mark-pos   .value_required:n = true,
2320         mark-sep   .dim_set:N  = \l__enumext_mark_sym_sep_dim,
2321         mark-sep   .value_required:n = true,
2322         show-ans   .bool_set:N = \l__enumext_show_answer_bool,
2323         show-ans   .initial:n  = false,
2324         show-ans   .value_required:n = true,
2325         show-pos   .bool_set:N = \l__enumext_show_position_bool,
2326         show-pos   .initial:n  = false,
2327         show-pos   .value_required:n = true,
2328         mark-ref   .tl_set:N   = \l__enumext_mark_ref_sym_tl,
2329         mark-ref   .initial:n  = \textreferencemark,
2330         mark-ref   .value_required:n = true,
2331         save-ref   .bool_set:N = \l__enumext_store_ref_key_bool,
2332         save-ref   .initial:n  = false,
2333         save-ref   .value_required:n = true,
2334       }
2335   }
2336 \clist_map_inline:nn { level-1, enumext* } { \__enumext_tmp:n {#1} }
```

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

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

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

```
2337 \cs_set_protected:Npn \__enumext_tmp:n #1
2338   {
2339     \keys_define:nn { enumext / #1 }
2340       {
2341         mark-ans   .tl_set:N  = \l__enumext_mark_answer_sym_tl,
2342         mark-ans   .initial:n = \textasteriskcentered,
2343         mark-ans   .value_required:n = true,
2344         mark-pos / left    .code:n = \str_set:Nn \l__enumext_mark_position_str { l },
2345         mark-pos / right   .code:n = \str_set:Nn \l__enumext_mark_position_str { r },
2346         mark-pos / center  .code:n = \str_set:Nn \l__enumext_mark_position_str { c },
2347         mark-pos / unknown .code:n =
2348                         \msg_error:nneee { enumext } { unknown-choice }
2349                           { mark-pos } { left, ~ right, ~ center } { \exp_not:n {##1} },
2350         mark-pos .initial:n = right,
2351         mark-pos .value_required:n = true,
2352         mark-pos .value_required:n  = true,
2353         mark-sep .dim_set:N  = \l__enumext_mark_sym_sep_dim,
2354         mark-sep .value_required:n = true,
2355         save-sep .tl_set:N   = \l__enumext_store_keyans_item_opt_sep_tl,
2356         save-sep .value_required:n = true,
2357         wrap-opt .cs_set_protected:Np = \__enumext_keyans_wrapper_opt:n ##1,
2358         wrap-opt .value_required:n = true,
2359         wrap-key .cs_set_protected:Np = \__enumext_keyans_wrapper_item:n ##1,
2360         wrap-key .value_required:n = true,
2361         show-ans .bool_set:N = \l__enumext_show_answer_bool,
2362         show-ans .initial:n  = false,
2363         show-ans .value_required:n = true,
2364         show-pos .bool_set:N = \l__enumext_show_position_bool,
2365         show-pos .initial:n  = false,
2366         show-pos .value_required:n = true,
2367       }
2368   }
2369 \clist_map_inline:nn { keyans, keyans* } { \__enumext_tmp:n {#1} }
```

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

13.27.1 Storing structure of the environments

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

__enumext_store_active_keys:n
__enumext_store_active_keys_vii:n

The functions `\__enumext_store_active_keys:n` and `\__enumext_store_active_keys_vii:n` will be responsible for the *"storing keys"* filtered from the *optional argument* of the environment in which the key `save-ans` is executed and the levels within this for the `enumext` and `enumext*` environments. We will execute this function only if the variable `\l__enumext_store_save_key_X_bool` is false, that is, the key `store-key` is not active, establishing the variable `\l__enumext_store_save_key_X_tl` with the filtered ⟨*keys*⟩.

```
2370 \cs_new_protected:Npn \__enumext_store_active_keys:n #1
2371   {
2372     \bool_if:cF { l__enumext_store_save_key_ \__enumext_level: _bool }
2373       {
2374         \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2375         \tl_set:ce
2376           { l__enumext_store_save_key_ \__enumext_level: _tl }
2377           { \__enumext_filter_save_key:n {#1} }
2378       }
2379   }
2380 \cs_new_protected:Npn \__enumext_store_active_keys_vii:n #1
2381   {
2382     \bool_if:NF \l__enumext_store_save_key_vii_bool
2383       {
2384         \tl_clear:N \l__enumext_store_save_key_vii_tl
2385         \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2386       }
2387   }
```

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

13.27.2 Setting save-key key

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

save-key

The values set by this key passed in the *optional argument* of the `enumext` and `enumext*` environments will override the values of the `\l__enumext_store_save_key_X_tl` variable set by the functions `\__enumext_store_active_keys:n` and `\__enumext_store_active_keys_vii:n`. Now define the key `save-key` for all levels of `enumext` and `enumext*` environments.

```
2388 \cs_set_protected:Npn \__enumext_tmp:n #1
2389   {
2390     \keys_define:nn { enumext / enumext* }
2391       {
2392         save-key .code:n = \__enumext_parse_save_key_vii:n {##1},
2393         save-key .value_required:n = true,
2394       }
2395     \keys_define:nn { enumext / #1 }
2396       {
2397         save-key .code:n = \__enumext_parse_save_key:n {##1},
2398         save-key .value_required:n = true,
2399       }
2400   }
2401 \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 keys"* in the variable `\l__enumext_store_save_key_X_tl` for `enumext` and `enumext*`.

```
2402 \cs_new_protected:Npn \__enumext_parse_save_key:n #1
2403   {
2404     \bool_set_true:c { l__enumext_store_save_key_ \__enumext_level: _bool }
2405     \tl_clear:c { l__enumext_save_key_ \__enumext_level: _tl }
2406     \tl_set:ce
2407       { l__enumext_store_save_key_ \__enumext_level: _tl }
2408       { \__enumext_filter_save_key:n {#1} }
2409   }
2410 \cs_new_protected:Npn \__enumext_parse_save_key_vii:n #1
```

```
2411    {
2412      \bool_set_true:N \l__enumext_store_save_key_vii_bool
2413      \tl_clear:N \l__enumext_store_save_key_vii_tl
2414      \tl_set:Ne \l__enumext_store_save_key_vii_tl { \__enumext_filter_save_key:n {#1} }
2415    }
```

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

13.27.3 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 keys"* we want to *stored* in *sequence* where {#1} represents the *optional argument* passed to the environment.

```
2416  \cs_new:Npn \__enumext_filter_save_key:n #1
2417    {
2418      \use:e
2419        {
2420          \keyval_parse:NNn
2421            \__enumext_filter_save_key_key:n
2422            \__enumext_filter_save_key_pair:nn {#1}
2423        }
2424    }
```

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

```
2425  \cs_new:Npn \__enumext_filter_save_key_key:n #1
2426    {
2427      \str_case:nnF {#1}
2428        {
2429          { resume } {} { resume* } {} { no-store } {} { base-fix } {}
2430        }
2431        { , { \exp_not:n {#1} } }
2432    }
```

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

```
2433  \cs_new:Npn \__enumext_filter_save_key_pair:nn #1#2
2434    {
2435      \str_case:nnF {#1}
2436        {
2437          { series   } {} { resume     } {} { save-ans } {} { save-ref   } {}
2438          { save-key } {} { check-ans  } {} { show-ans } {} { show-pos   } {}
2439          { wrap-ans } {} { mark-ans   } {} { wrap-opt } {} { wrap-key } {}
2440          { save-sep } {} { mark-sep   } {} { mark-ref } {} { mini-env  } {}
2441          { mini-sep } {} { mini-right } {} { mini-right* } {}
2442        }
2443        { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
2444    }
```

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

13.27.4 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.

```
2445  \cs_new_protected:Npn \__enumext_store_addto_prop:n #1
2446    {
2447      \prop_gput_if_not_in:cen { g__enumext_ \l__enumext_store_name_tl _prop }
2448        {
2449          \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + 1 }
2450        }
2451        { #1 }
2452    }
2453  \cs_generate_variant:Nn \__enumext_store_addto_prop:n { V }
```

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

13.27.5 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.

```
2454 \cs_new_protected:Npn \__enumext_store_addto_seq:n #1
2455   {
2456     \seq_gput_right:cn { g__enumext_ \l__enumext_store_name_tl _seq } { #1 }
2457   }
2458 \cs_generate_variant:Nn \__enumext_store_addto_seq:n { v, V }
```

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

13.27.6 Functions for storing structure in the sequence

__enumext_store_level_open:
__enumext_store_level_close:

The *"storing structure"* is handled by the functions __enumext_store_level_open: and __enumext_-store_level_close: which are executed per level within the enumext environment.

```
2459 \cs_new_protected:Nn \__enumext_store_level_open:
2460   {
2461     \bool_if:NT \l__enumext_check_answers_bool
2462       {
2463         \tl_if_empty:cTF { l__enumext_store_save_key_ \__enumext_level: _tl }
2464           {
2465             \__enumext_store_addto_seq:n
2466               {
2467                 \item \begin{enumext}
2468               }
2469           }
2470           {
2471             \tl_put_left:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2472               {
2473                 \item \begin{enumext} [
2474               }
2475             \tl_put_right:cn { l__enumext_store_save_key_ \__enumext_level: _tl }
2476               {
2477                 ]
2478               }
2479             \__enumext_store_addto_seq:v { l__enumext_store_save_key_ \__enumext_level: _tl }
2480           }
2481       }
2482   }
2483 \cs_new_protected:Nn \__enumext_store_level_close:
2484   {
2485     \bool_if:NT \l__enumext_check_answers_bool
2486       {
2487         \__enumext_store_addto_seq:n { \end{enumext} }
2488       }
2489   }
```

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

__enumext_store_level_open_vii:
__enumext_store_level_close_vii:

The *"storing structure"* is handled by the functions __enumext_store_level_open_vii: and __enumext_-store_level_close_vii: which are executed in the enumext* environment.

```
2490 \cs_new_protected:Nn \__enumext_store_level_open_vii:
2491   {
2492     \bool_if:NT \l__enumext_check_answers_bool
2493       {
2494         \tl_if_empty:NTF \l__enumext_store_save_key_vii_tl
2495           {
2496             \__enumext_store_addto_seq:n
2497               {
2498                 \item \begin{enumext*}
2499               }
2500           }
2501           {
2502             \tl_put_left:Nn \l__enumext_store_save_key_vii_tl
2503               {
2504                 \item \begin{enumext*}[
2505               }
```

```
2506                \tl_put_right:Nn \l__enumext_store_save_key_vii_tl
2507                  {
2508                    ]
2509                  }
2510                \__enumext_store_addto_seq:V \l__enumext_store_save_key_vii_tl
2511              }
2512          }
2513      }
2514  \cs_new_protected:Nn \__enumext_store_level_close_vii:
2515    {
2516      \bool_if:NT \l__enumext_check_answers_bool
2517        {
2518          \__enumext_store_addto_seq:n { \end{enumext*} }
2519        }
2520    }
```

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

13.27.7 Function for show marks and position

__enumext_print_keyans_box:NN
__enumext_print_keyans_box:cc

The function __enumext_print_keyans_box:NN print a box in the left margin with \l__enumext_mark_-answer_sym_tl used by the wrap-ans, show-ans and show-pos keys. The function takes two arguments:

#1 : \l__enumext_labelwidth_X_dim
#2 : \l__enumext_labelsep_X_dim

```
2521  \cs_new_protected:Nn \__enumext_print_keyans_box:NN
2522    {
2523      \mode_leave_vertical:
2524      \skip_horizontal:n { -\dim_use:N #2 }
2525      \hbox_overlap_left:n
2526        {
2527          \makebox[ \dim_use:N #1 ][ \l__enumext_mark_position_str ]
2528            {
2529              \tl_use:N \l__enumext_mark_answer_sym_tl
2530            }
2531        }
2532      \skip_horizontal:n { \dim_use:N #2 }
2533    }
2534  \cs_generate_variant:Nn \__enumext_print_keyans_box:NN { cc }
```

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

13.28 The internal label and ref

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

__enumext_store_internal_ref:

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

```
2535  \cs_new_protected:Nn \__enumext_store_internal_ref:
2536    {
2537      \cs_set_protected:Npn \__enumext_tmp:n ##1
2538        {
2539          \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
2540          \tl_reverse:c { l__enumext_label_copy_##1_tl }
2541          \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
2542          \tl_reverse:c { l__enumext_label_copy_##1_tl }
2543        }
2544      \clist_map_inline:nn { i, ii, iii, iv, vii } { \__enumext_tmp:n {##1} }
2545      \cs_set:Npn \__enumext_tmp:n ##1
2546        { . \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.

```
2547      \bool_lazy_all:nT
2548        {
2549          { \bool_if_p:N \g__enumext_starred_bool }
2550          { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2551        }
2552        {
2553          \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2554            { \tl_use:N \l__enumext_label_copy_vii_tl }
```

```
2555              }
2556          \bool_lazy_all:nT
2557              {
2558                { \bool_not_p:n { \g__enumext_standar_bool } }
2559                { \bool_if_p:N \l__enumext_standar_bool }
2560                { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2561              }
2562              {
2563                \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2564                  {
2565                    \tl_use:N \l__enumext_label_copy_vii_tl
2566                    \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2567                  }
2568              }
```

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.

```
2569          \bool_lazy_all:nT
2570              {
2571                { \bool_if_p:N \g__enumext_standar_bool }
2572                { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2573                { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2574              }
2575              {
2576                \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2577                  {
2578                    \tl_use:N \l__enumext_label_copy_i_tl
2579                    \int_step_function:nnN { 2 } { \l__enumext_level_int } \__enumext_tmp:n
2580                  }
2581              }
2582          \cs_set:Npn \__enumext_tmp:n ##1
2583            { \tl_use:c { l__enumext_label_copy_ \int_to_roman:n {##1} _tl } . }
2584          \bool_lazy_all:nT
2585              {
2586                { \bool_if_p:N \g__enumext_standar_bool }
2587                { \bool_if_p:N \l__enumext_starred_bool }
2588                { \int_compare_p:nNn { \l__enumext_level_int } > { 0 } }
2589              }
2590              {
2591                \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
2592                  {
2593                    \int_step_function:nnN { 1 } { \l__enumext_level_int } \__enumext_tmp:n
2594                    \tl_use:N \l__enumext_label_copy_vii_tl
2595                  }
2596              }
```

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

```
2597          \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
2598              {
2599                \l__enumext_store_name_tl \c_colon_str
2600                \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
2601              }
```

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.

```
2602          \tl_put_right:Ne \l__enumext_write_aux_file_tl
2603              {
2604                \__enumext_newlabel:nn
2605                  { \exp_not:V \l__enumext_newlabel_arg_one_tl }
2606                  { \l__enumext_newlabel_arg_two_tl }
2607              }
2608          \l__enumext_write_aux_file_tl
2609      }
```

(*End of definition for* __enumext_store_internal_ref:.)

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

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

```
2610  \cs_new_protected:Npn \__enumext_store_anskey_code:n #1
2611    {
2612      \int_gincr:N \g__enumext_item_anskey_int
2613      \__enumext_store_addto_prop:n {#1}
2614      \bool_if:NT \l__enumext_store_ref_key_bool
2615        {
2616          \__enumext_store_internal_ref:
2617        }
2618      \__enumext_anskey_show_wrap_left:n { #1 }
```

Now we start processing the $[\langle key = val \rangle]$ 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 $\langle keys \rangle$, if the `break-col` key is present and the command is running under enumext (not in enumext*) we will add `\columnbreak` and then `\item`.

```
2619      \tl_clear:N \l__enumext_store_anskey_arg_tl
2620      \bool_lazy_and:nnT
2621        { \bool_if_p:N \l__enumext_store_columns_break_bool }
2622        { \bool_not_p:n { \l__enumext_starred_bool } }
2623        {
2624          \tl_put_left:Nn \l__enumext_store_anskey_arg_tl { \columnbreak }
2625        }
2626      \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 ($\langle number \rangle$) to `\l__enumext_store_anskey_arg_tl`.

```
2627      \bool_lazy_and:nnT
2628        { \bool_not_p:n { \l__enumext_starred_bool } }
2629        { \int_compare_p:nNn { \l__enumext_store_item_join_int } > { 1 } }
2630        {
2631          \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2632            {
2633              ( \exp_not:V \l__enumext_store_item_join_int )
2634            }
2635        }
```

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 {$\langle argument \rangle$} for `\anskey` or $\langle body \rangle$ for anskey*.

```
2636      \bool_if:NTF \l__enumext_store_item_star_bool
2637        {
2638          \tl_put_right:Nn \l__enumext_store_anskey_arg_tl { * }
2639          \tl_if_empty:NF \l__enumext_store_item_symbol_tl
2640            {
2641              \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2642                {
2643                  [ \exp_not:V \l__enumext_store_item_symbol_tl ]
2644                }
2645            }
2646          \dim_compare:nT
2647            {
2648              \l__enumext_store_item_symbol_sep_dim != \c_zero_dim
2649            }
2650            {
2651              \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2652                {
2653                  [ \exp_not:V \l__enumext_store_item_symbol_sep_dim ]
2654                }
2655            }
2656          \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2657        }
2658        {
2659          \tl_put_right:Nn \l__enumext_store_anskey_arg_tl {#1}
2660        }
```

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*.

```
2661      \bool_lazy_and:nnT
2662        { \bool_if_p:N \l__enumext_store_ref_key_bool }
2663        { \bool_if_p:N \l__enumext_hyperref_bool }
2664        {
2665          \tl_put_right:Ne \l__enumext_store_anskey_arg_tl
2666            {
2667              \hfill \exp_not:N \hyperlink { \exp_not:V \l__enumext_newlabel_arg_one_tl }
```

```
2668                    { \exp_not:V \l__enumext_mark_ref_sym_tl }
2669                  }
2670              }
2671          \__enumext_store_addto_seq:V \l__enumext_store_anskey_arg_tl
2672      }
```

(*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.

```
2673  \cs_new_protected:Npn \__enumext_anskey_show_wrap_arg:n #1
2674    {
2675      \par
2676      \bool_if:NTF \l__enumext_starred_bool
2677        {
2678          \__enumext_print_keyans_box:NN
2679            \l__enumext_labelwidth_vii_dim  \l__enumext_labelsep_vii_dim
2680        }
2681        {
2682          \__enumext_print_keyans_box:cc
2683            { l__enumext_labelwidth_ \__enumext_level: _dim }
2684            { l__enumext_labelsep_ \__enumext_level: _dim }
2685        }
2686      \__enumext_anskey_wrapper:n { #1 }
2687    }
```

(*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.

```
2688  \cs_new_protected:Npn \__enumext_anskey_show_wrap_left:n #1
2689    {
2690      \bool_if:NT \l__enumext_show_answer_bool
2691        {
2692          \__enumext_anskey_show_wrap_arg:n { #1 }
2693        }
2694      \bool_if:NT \l__enumext_show_position_bool
2695        {
2696          \tl_set:Ne \l__enumext_mark_answer_sym_tl
2697            {
2698              \group_begin:
2699              \exp_not:N \normalfont
2700              \exp_not:N \footnotesize [ \int_eval:n
2701                {
2702                  \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
2703                }
2704                ]
2705              \group_end:
2706            }
2707          \__enumext_anskey_show_wrap_arg:n { #1 }
2708        }
2709    }
```

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

13.30   The command \anskey

Since we will be *"storing content"* in a list environment within *sequences* and can (more or less) manage the options passed to each level, it is necessary that we have a little more control over \item when storing.

The \anskey command will cover this point and give it similar behaviour to that of \item in the enumext and enumext* environments executed as follows \anskey[⟨*key = val*⟩]{⟨*content*⟩}.

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

```
2710  \keys_define:nn { enumext / anskey }
2711    {
2712      break-col .bool_set:N = \l__enumext_store_columns_break_bool,
2713      break-col .default:n  = true,
2714      break-col .value_forbidden:n = true,
2715      item-join .int_set:N  = \l__enumext_store_item_join_int,
```

```
2716        item-join .value_required:n = true,
2717        item-star .bool_set:N = \l__enumext_store_item_star_bool,
2718        item-star .default:n  = true,
2719        item-star .value_forbidden:n = true,
2720        item-sym* .tl_set:N   = \l__enumext_store_item_symbol_tl,
2721        item-sym* .value_required:n = true,
2722        item-pos* .dim_set:N  = \l__enumext_store_item_symbol_sep_dim,
2723        item-pos* .value_required:n = true,
2724        unknown   .code:n     = { \__enumext_anskey_unknown:n {#1} },
2725      }
```

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.

```
2726  \cs_new_protected:Npn \__enumext_anskey_unknown:n #1
2727      {
2728        \exp_args:NV \__enumext_anskey_unknown:nn \l_keys_key_str {#1}
2729      }
2730  \cs_new_protected:Npn \__enumext_anskey_unknown:nn #1 #2
2731      {
2732        \tl_if_blank:nTF {#2}
2733          {
2734            \msg_error:nnn { enumext } { anskey-cmd-key-unknown } {#1}
2735          }
2736          {
2737            \msg_error:nnnn { enumext } { anskey-cmd-key-value-unknown } {#1} {#2}
2738          }
2739      }
```

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

🫗* The \anskey command will only be present when using the save-ans key in enumext and enumext* environments, otherwise it will return an error.

\anskey   We will first call the function __enumext_anskey_safe_outer: to be sure where we execute the command, then we will check the state of the variable \l__enumext_check_answers_bool set by the key no-store, if is true we will increment \g__enumext_item_anskey_int for the internal *"check answer"* system and execute the function __enumext_anskey_safe_inner:n to ensure that the command is not nested and that the argument is not empty, finally search the [⟨*key = val*⟩] and call the function __enumext_store_-anskey_code:n.

```
2740  \NewDocumentCommand \anskey { o +m }
2741      {
2742        \__enumext_anskey_safe_outer:
2743        \group_begin:
2744          \bool_if:NT \l__enumext_check_answers_bool
2745            {
2746              \tl_if_novalue:nF {#1}
2747                {
2748                  \keys_set:nn { enumext / anskey } {#1}
2749                }
2750              \tl_if_blank:nTF {#2}
2751                {
2752                  \msg_error:nn { enumext } { anskey-empty-arg }
2753                }
2754                {
2755                  \__enumext_anskey_safe_inner:
2756                  \__enumext_store_anskey_code:n {#2}
2757                }
2758            }
2759        \group_end:
2760      }
```

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

13.30.1   Internal functions for the command

__enumext_anskey_safe_outer: The __enumext_store_anskey_safe_outer: function will return the appropriate messages when the
__enumext_anskey_safe_inner: command is executed outside the environment in which the save-ans key was activated.

```
2761  \cs_new_protected:Nn \__enumext_anskey_safe_outer:
2762      {
2763        \bool_if:NF \l__enumext_store_active_bool
2764          {
2765            \msg_error:nnnn { enumext } { anskey-wrong-place }{ anskey }{ enumext }
2766          }
```

```
2767    \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2768      {
2769        \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans }
2770      }
2771    \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2772      {
2773        \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyans* }
2774      }
2775    \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2776      {
2777        \msg_error:nnnn { enumext } { command-wrong-place }{ anskey }{ keyanspic }
2778      }
2779    }
```

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.

```
2780  \cs_new_protected:Nn \__enumext_anskey_safe_inner:
2781    {
2782      \int_incr:N \l__enumext_anskey_level_int
2783      \int_compare:nNnT { \l__enumext_anskey_level_int } > { 1 }
2784        {
2785          \msg_error:nn { enumext } { anskey-nested }
2786        }
2787      \bool_if:NF \l__enumext_item_number_bool
2788        {
2789          \msg_error:nn { enumext } { anskey-unnumber-item }
2790        }
2791      \mode_if_math:T
2792        {
2793          \msg_error:nne { enumext } { anskey-math-mode } { \c_backslash_str anskey }
2794        }
2795    }
```

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

13.31   The environment anskey*

Managing *verbatim content* in an environment is quite complicated, I learned that when creating the scontents package, so to be able to have support at this point it is best to play a little with the internal code of scontents and *hooks*. Some considerations I should have here before implementing this:

- If some package, class or user has defined the environment with the same name somewhere in the document it would be a problem, you would not know what argument has been passed to store-env, if you are using the key print-env or the write-out key, sure, I can detect and modify it within the enumext and enumext* environments, but it would look strange not to have some keys available when running within these environments.
- A better (perhaps a bit paranoid) option is to define it within the environment in which the save-ans key is executed. and have it available only when that key is executed, here I would have absolute control of the ⟨keys⟩ and I make sure that write-out is not used, then using *hooks after* I undefine it and using *hook before* I check if it has been created by any package, class or user and I return a error, then the user will have to see how to solve the problem.

__enumext_undefine_anskey_env: The function __enumext_undefine_anskey_env: will undefine the environment anskey* and will be passed to the function __enumext_execute_after_env: (§13.32) which is executed after the environment in which the key save-ans is active.

```
2796  \cs_new_protected:Nn \__enumext_undefine_anskey_env:
2797    {
2798      \cs_undefine:c { anskey* }
2799      \cs_undefine:c { endanskey* }
2800      \cs_undefine:c { __scontents_anskey*_env_begin: }
2801      \cs_undefine:c { __scontents_anskey*_env_end: }
2802    }
```

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

```
2803  \__enumext_before_env:nn { enumext }
2804    {
2805      \bool_lazy_and:nnT
2806        { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2807        { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2808        {
2809          \cs_if_free:cF { __scontents_anskey*_env_begin: }
2810            {
```

```
2811                  \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2812              }
2813          }
2814      }
2815  \__enumext_before_env:nn { enumext* }
2816      {
2817      \bool_lazy_and:nnT
2818          { \int_compare_p:nNn { \l__enumext_level_int } = { 0 } }
2819          { \int_compare_p:nNn { \l__enumext_level_h_int } = { 0 } }
2820          {
2821              \cs_if_free:cF { __scontents_anskey*_env_begin: }
2822                  {
2823                      \msg_error:nnn { enumext } { anskey-env-error } { anskey* }
2824                  }
2825          }
2826      }
```

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.

```
2827  \__enumext_before_env:nn { anskey* }
2828      {
2829      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
2830          {
2831              \msg_error:nnn { enumext } { anskey-env-wrong }{ keyans }
2832          }
2833      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
2834          {
2835              \msg_error:nnn { enumext } { anskey-env-wrong } { keyans* }
2836          }
2837      \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
2838          {
2839              \msg_error:nnn { enumext } { anskey-env-wrong } { keyanspic }
2840          }
2841      \bool_if:NF \l__enumext_item_number_bool
2842          {
2843              \msg_error:nn { enumext } { anskey-unnumber-item }
2844          }
2845      \mode_if_math:T
2846          {
2847              \msg_error:nnn { enumext } { anskey-math-mode } { anskey* }
2848          }
2849      }
```

(*End of definition for* __enumext_undefine_anskey_env:.)

anskey*
__enumext_anskey_env_make:n
__enumext_anskey_env_make:V
__enumext_anskey_env_define_keys:
__enumext_anskey_env_unknown:n
__enumext_anskey_env_unknown:nn
__enumext_anskey_env_reset_keys:
__enumext_rescan_anskey_env:n

The function __enumext_anskey_env_make:n creates the environment anskey* (*custom version* of scontents environment) by setting the initial keys store-env={⟨*store name*⟩} and print-env=false. To maintain the *scope* of the environment and that it is only active when the key save-ans is active we will pass this function to the function __enumext_storing_exec: (§13.26.1) and we will execute it only if the variable \l__enumext_anskey_env_bool is true, with this we prevent it from being executed again when the environment is nested and the key save-ans is active, which returns an error for part of the package scontents.

```
2850  \cs_new_protected:Npn \__enumext_anskey_env_make:n #1
2851      {
2852      \bool_if:NT \l__enumext_anskey_env_bool
2853          {
2854              \newenvsc{anskey*}[store-env=#1,print-env=false]
2855              \__enumext_anskey_env_exec:
2856          }
2857      }
2858  \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.

```
2859  \cs_new_protected:Nn \__enumext_anskey_env_define_keys:
2860      {
2861      \keys_define:nn { scontents / scontents }
2862          {
2863              break-col .bool_gset:N = \g__enumext_store_columns_break_bool,
2864              break-col .default:n   = true,
```

```
2865          break-col .value_forbidden:n = true,
2866          item-join .int_gset:N  = \g__enumext_store_item_join_int,
2867          item-join .value_required:n = true,
2868          item-star .bool_gset:N = \g__enumext_store_item_star_bool,
2869          item-star .default:n   = true,
2870          item-star .value_forbidden:n = true,
2871          item-sym* .tl_gset:N   = \g__enumext_store_item_symbol_tl,
2872          item-sym* .value_required:n = true,
2873          item-pos* .dim_gset:N  = \g__enumext_store_item_symbol_sep_dim,
2874          item-pos* .value_required:n = true,
2875          print-env .undefine:,
2876          store-env .undefine:,
2877          write-out .undefine:,
2878          unknown   .code:n      = { \__enumext_anskey_env_unknown:n {##1} },
2879        }
2880    }
```

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.

```
2881  \cs_new_protected:Npn \__enumext_anskey_env_unknown:n #1
2882    {
2883      \exp_args:NV \__enumext_anskey_env_unknown:nn \l_keys_key_str {#1}
2884    }
2885  \cs_new_protected:Npn \__enumext_anskey_env_unknown:nn #1#2
2886    {
2887      \tl_if_blank:nTF {#2}
2888        {
2889          \msg_error:nnn { enumext } { anskey-env-key-unknown } {#1}
2890        }
2891        {
2892          \msg_error:nnnn { enumext } { anskey-env-key-value-unknown } {#1} {#2}
2893        }
2894    }
```

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.

```
2895  \cs_new_protected:Nn \__enumext_anskey_env_reset_keys:
2896    {
2897      \keys_define:nn { scontents / scontents }
2898        {
2899          break-col .undefine:,
2900          item-join .undefine:,
2901          item-star .undefine:,
2902          item-sym* .undefine:,
2903          item-pos* .undefine:,
2904          write-out .code:n     = {
2905                                    \bool_set_false:N \l__scontents_storing_bool
2906                                    \bool_set_true:N  \l__scontents_writing_bool
2907                                    \tl_set:Nn \l__scontents_fname_out_tl {##1}
2908                                  },
2909          write-out .value_required:n = true,
2910          print-env .meta:nn  = { scontents } { print-env = ##1 },
2911          print-env .default:n = true,
2912          store-env .meta:nn  = { scontents } { store-env = ##1 },
2913          unknown   .code:n   = { \__scontents_parse_environment_keys:n {##1} },
2914        }
2915    }
```

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*.

```
2916  \cs_new_protected:Npn \__enumext_rescan_anskey_env:n #1
2917    {
2918      \group_begin:
2919        \int_set:Nn \tex_newlinechar:D { `\^^J }
2920        \__scontents_rescan_tokens:x
2921          {
2922            \endgroup % This assumes \catcode`\\=0... Things might go off otherwise.
2923            #1
2924          }
2925    }
```

__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*⟩.

```
2926  \cs_new_protected:Nn \__enumext_anskey_env_exec:
2927    {
2928      \__enumext_before_env:nn { anskey* }
2929        {
2930          \__enumext_anskey_env_define_keys:
2931        }
```

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.

```
2932      \hook_if_empty:nF {env/anskey*/after}
2933        {
2934          \hook_gremove_code:nn {env/anskey*/after} { * }
2935        }
2936      \__enumext_after_env:nn { anskey* }
2937        {
2938          \__enumext_anskey_env_save_keys:
2939          \tl_clear:N \l__enumext_store_anskey_env_tl
2940          \tl_clear:N \l__enumext_store_anskey_opt_tl
2941          \bool_if:NT \l__enumext_check_answers_bool
2942            {
2943              \tl_gset:Ne \l__enumext_store_anskey_env_tl
2944                {
2945                  \seq_item:ce { g__scontents_name_ \l__enumext_store_name_tl _seq } { -1 }
2946                }
2947              \regex_match:nVTF
2948                { ^\s* \z | ^\s* \u{c__scontents_hidden_space_str} \z }
2949                \l__enumext_store_anskey_env_tl
2950                {
2951                  \msg_error:nn { enumext } { anskey-empty-arg }
2952                }
2953                {
2954                  \__enumext_anskey_env_store:
2955                }
2956            }
2957          \__enumext_anskey_env_clean_vars:
2958          \__enumext_anskey_env_reset_keys:
2959        }
2960    }
```

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

(*End of definition for* __enumext_anskey_env_exec:.)

__enumext_anskey_env_save_keys:
__enumext_anskey_env_store:
__enumext_anskey_env_clean_vars:

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

```
2961  \cs_new_protected:Nn \__enumext_anskey_env_save_keys:
2962    {
2963      \bool_lazy_and:nnT
2964        { \bool_if_p:N \g__enumext_store_columns_break_bool }
2965        { \bool_not_p:n { \l__enumext_starred_bool } }
2966        {
2967          \tl_put_left:Ne \l__enumext_store_anskey_opt_tl { ,break-col, }
2968        }
```

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

```
2969      \bool_lazy_and:nnT
2970        { \bool_not_p:n { \l__enumext_starred_bool } }
2971        { \int_compare_p:nNn { \g__enumext_store_item_join_int } > { 1 } }
2972        {
2973          \tl_put_left::Ne \l__enumext_store_anskey_opt_tl
2974            {
2975              ,item-join = \exp_not:V \g__enumext_store_item_join_int,
2976            }
2977        }
```

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

```
2978      \bool_if:NT \g__enumext_store_item_star_bool
2979        {
2980          \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2981            {
2982              ,item-star,
2983            }
2984          \tl_if_empty:NF \g__enumext_store_item_symbol_tl
2985            {
2986              \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2987                {
2988                  ,item-sym* = \exp_not:V \g__enumext_store_item_symbol_tl,
2989                }
2990            }
2991          \dim_compare:nT
2992            {
2993              \g__enumext_store_item_symbol_sep_dim != \c_zero_dim
2994            }
2995            {
2996              \tl_put_left:Ne \l__enumext_store_anskey_opt_tl
2997                {
2998                  ,item-pos* = \exp_not:V \g__enumext_store_item_symbol_sep_dim,
2999                }
3000            }
3001        }
3002      }
```

The function `\__enumext_anskey_env_store:` will be responsible for storing the content of the environment using the functions `\__enumext_store_anskey_code:n` and `\__enumext_rescan_anskey_env:n`.

```
3003  \cs_new_protected:Nn \__enumext_anskey_env_store:
3004    {
3005      \group_begin:
3006        \tl_if_empty:NTF \l__enumext_store_anskey_opt_tl
3007          {
3008            \exp_args:Ne
3009              \__enumext_store_anskey_code:n
3010                {
3011                  \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
3012                }
3013          }
3014          {
3015            \keys_set_known:nV { enumext / anskey } \l__enumext_store_anskey_opt_tl
3016            \exp_args:Ne
3017              \__enumext_store_anskey_code:n
3018                {
3019                  \__enumext_rescan_anskey_env:n { \l__enumext_store_anskey_env_tl }
3020                }
3021          }
3022      \group_end:
3023    }
```

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

```
3024  \cs_new_protected:Nn \__enumext_anskey_env_clean_vars:
3025    {
3026      \bool_gset_false:N \g__enumext_store_columns_break_bool
3027      \int_gzero:N       \g__enumext_store_item_join_int
3028      \bool_gset_false:N \g__enumext_store_item_star_bool
3029      \tl_gclear:N       \g__enumext_store_item_symbol_tl
3030      \dim_gzero:N       \g__enumext_store_item_symbol_sep_dim
3031    }
```

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

13.32 Executing anskey*, check-ans and write .log

`\__enumext_execute_after_env:`

The `\__enumext_execute_after_env:` function will first return the appropriate message for the end of the environment in which the `save-ans` key is being executed, then call the `\__enumext_item_answer_diff:` function and then will write the values of the global variables used to the `.log` file. If the key `check-ans` is active it will execute the function `\__enumext_check_ans_show:` and show the result in the terminal,

otherwise it will execute the function __enumext_check_ans_log: and write the results in the .log file, undefine the environment anskey* (§13.31) through the function __enumext_undefine_anskey_env: and finally we execute the function __enumext_reset_global_vars: returning the used variables to their original state.

```
3032  \cs_new_protected:Nn \__enumext_execute_after_env:
3033    {
3034      \int_compare:nNnT { \l__enumext_level_int } = { 0 }
3035        {
3036          \tl_if_empty:NF \g__enumext_store_name_tl
3037            {
3038              \__enumext_stop_save_ans_msg:
3039              \__enumext_item_answer_diff:
3040              \__enumext_log_global_vars:
3041              \__enumext_log_answer_vars:
3042              \bool_if:NTF \g__enumext_check_ans_key_bool
3043                {
3044                  \__enumext_check_ans_show:
3045                }
3046                { \__enumext_check_ans_log: }
3047              \__enumext_undefine_anskey_env:
3048            }
3049          \__enumext_reset_global_vars:
3050        }
3051    }
```

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

🎇 This function is passed to the function __enumext_after_env:nn for the environments enumext (§13.39) and enumext* (§13.44) and it is executed only when the environments are not nested or at some level of these..

13.33   Common functions for keyans, keyans* and keyanspic

13.33.1   Storing content in prop list

__enumext_keyans_addto_prop:n

The function __enumext_keyans_addto_prop:n will pass the the current ⟨*label*⟩ for \item* in keyans environment and the current ⟨*label*⟩ for \anspic* in keyanspic environment followed by the ⟨*contents*⟩ of the *optional argument* of both commands to the \l__enumext_store_current_label_tl variable, which will be stored to the *prop list* defined by the save-ans key using the function __enumext_store_addto_-prop:V.

```
3052  \cs_new_protected:Npn \__enumext_keyans_addto_prop:n #1
3053    {
3054      \tl_clear:N \l__enumext_store_current_label_tl
3055      \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
3056        {
3057          \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_vi_tl }
3058        }
3059        {
3060          \tl_put_right:Ne \l__enumext_store_current_label_tl { \l__enumext_label_v_tl }
3061        }
```

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

```
3062      \tl_if_novalue:nF { #1 }
3063        {
3064          \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
3065            {
3066              \tl_put_right:Ne \l__enumext_store_current_label_tl
3067                {
3068                  \l__enumext_store_keyans_item_opt_sep_tl
3069                }
3070            }
3071          \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
3072        }
3073      \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
3074    }
```

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

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

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

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

```
3075 \cs_new_protected:Nn \__enumext_keyans_store_ref:
3076   {
3077     \bool_if:NT \l__enumext_store_ref_key_bool
3078       {
3079         \cs_set_protected:Npn \__enumext_tmp:n ##1
3080           {
3081             \tl_set_eq:cc { l__enumext_label_copy_##1_tl } { l__enumext_label_##1_tl }
3082             \tl_reverse:c { l__enumext_label_copy_##1_tl }
3083             \tl_remove_once:cn { l__enumext_label_copy_##1_tl } { . }
3084             \tl_reverse:c { l__enumext_label_copy_##1_tl }
3085           }
3086         \clist_map_inline:nn { i, v, vi, vii, viii } { \__enumext_tmp:n {##1} }
3087         \__enumext_keyans_store_ref_aux_i:
3088       }
3089   }
```

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.

```
3090 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_i:
3091   {
3092     \bool_if:NT \g__enumext_starred_bool
3093       {
3094         \tl_set_eq:NN \l__enumext_label_copy_i_tl \l__enumext_label_copy_vii_tl
3095       }
3096     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } = { 1 }
3097       {
3098         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
3099           { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_vi_tl }
3100       }
3101     \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
3102       {
3103         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
3104           { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_v_tl }
3105       }
3106     \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
3107       {
3108         \tl_put_right:Ne \l__enumext_newlabel_arg_two_tl
3109           { \l__enumext_label_copy_i_tl . \l__enumext_label_copy_viii_tl }
3110       }
3111     \tl_put_right:Ne \l__enumext_newlabel_arg_one_tl
3112       {
3113         \l__enumext_store_name_tl \c_colon_str
3114         \int_eval:n { \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } }
3115       }
3116     \__enumext_keyans_store_ref_aux_ii:
3117   }
```

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.

```
3118 \cs_new_protected:Nn \__enumext_keyans_store_ref_aux_ii:
3119   {
3120     \tl_put_right:Ne \l__enumext_write_aux_file_tl
3121       {
3122         \__enumext_newlabel:nn
3123           { \exp_not:V \l__enumext_newlabel_arg_one_tl }
3124           { \l__enumext_newlabel_arg_two_tl }
3125       }
3126     \l__enumext_write_aux_file_tl
3127   }
```

(*End of definition for* `\__enumext_keyans_store_ref:`, `\__enumext_keyans_store_ref_aux_i:`, *and* `\__enumext_keyans_-store_ref_aux_ii:`.)

13.33.3   Storing content in sequence

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.

```
3128 \cs_new_protected:Npn \__enumext_keyans_addto_seq:n #1
3129   {
3130     \tl_clear:N \l__enumext_store_current_label_tl
3131     \int_compare:nNnTF { \l__enumext_keyans_pic_level_int } = { 1 }
3132       {
3133         \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_vi_tl }
3134       }
3135       {
3136         \tl_put_right:Ne \l__enumext_store_current_label_tl { \item \l__enumext_label_v_tl }
3137       }
3138     \tl_if_novalue:nF { #1 }
3139       {
3140         \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
3141           {
3142             \tl_put_right:Ne \l__enumext_store_current_label_tl
3143               {
3144                 \l__enumext_store_keyans_item_opt_sep_tl
3145               }
3146           }
3147         \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
3148       }
3149     \__enumext_keyans_addto_seq_link:
3150   }
```

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.

```
3151 \cs_new_protected:Nn \__enumext_keyans_addto_seq_link:
3152   {
3153     \bool_lazy_and:nnT
3154       { \bool_if_p:N \l__enumext_store_ref_key_bool }
3155       { \bool_if_p:N \l__enumext_hyperref_bool }
3156       {
3157         \tl_put_right:Ne \l__enumext_store_current_label_tl
3158           {
3159             \hfill \exp_not:N \hyperlink
3160               {
3161                 \exp_not:V \l__enumext_newlabel_arg_one_tl
3162               }
3163               { \exp_not:V \l__enumext_mark_ref_sym_tl }
3164           }
3165       }
3166     \__enumext_store_addto_seq:V \l__enumext_store_current_label_tl
3167     \bool_if:NT \l__enumext_check_answers_bool
3168       {
3169         \int_gincr:N \g__enumext_item_anskey_int
3170       }
3171   }
```

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

13.33.4 The show-ans and show-pos keys for keyans and keyanspic

`\__enumext_keyans_save_item_opt:n`
`\__enumext_keyans_show_item_opt:`

The function `\__enumext_keyans_save_item_opt:n` will save the optional argument of `\item*` and `\anspic*` in the variable `\l__enumext_store_current_opt_arg_tl`.

```
3172 \cs_new_protected:Npn \__enumext_keyans_save_item_opt:n #1
3173   {
3174     \tl_if_novalue:nF { #1 }
3175       {
3176         \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
3177       }
3178   }
```

The function `\__enumext_keyans_show_item_opt:` will print the optional arguments of `\item*` and `\anspic*` when the `show-ans` or `show-pos` keys are set.

```
3179 \cs_new_protected:Nn \__enumext_keyans_show_item_opt:
3180   {
```

```
3181        \tl_if_empty:NF \l__enumext_store_current_opt_arg_tl
3182          {
3183            \bool_lazy_or:nnT
3184              { \bool_if_p:N \l__enumext_show_answer_bool }
3185              { \bool_if_p:N \l__enumext_show_position_bool }
3186              {
3187                \__enumext_keyans_wrapper_opt:n
3188                  { \l__enumext_store_current_opt_arg_tl } \c_space_tl
3189              }
3190          }
3191      }
```

(*End of definition for* __enumext_keyans_save_item_opt:n *and* __enumext_keyans_show_item_opt:.)

__enumext_keyans_pos_mark_set:
__enumext_keyans_show_ans:
__enumext_keyans_show_pos:

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

```
3192  \cs_new_protected:Nn \__enumext_keyans_pos_mark_set:
3193    {
3194      \__enumext_label_width_by_box:Nn
3195        \l__enumext_mark_sep_tmpa_dim { \l__enumext_label_v_tl }
3196      \str_case:Vn \l__enumext_align_label_pos_v_str
3197        {
3198          { l }
3199            {
3200              \dim_set:Nn \l__enumext_mark_sep_tmpb_dim { \c_zero_dim }
3201            }
3202          { r }
3203            {
3204              \dim_set:Nn \l__enumext_mark_sep_tmpb_dim
3205                { \l__enumext_labelwidth_v_dim - \l__enumext_mark_sep_tmpa_dim }
3206            }
3207          { c }
3208            {
3209              \dim_set:Nn \l__enumext_mark_sep_tmpb_dim
3210                { 0.5\l__enumext_labelwidth_v_dim - 0.5\l__enumext_mark_sep_tmpa_dim }
3211            }
3212        }
```

Here we set the default value for the key mark-sep.

```
3213      \dim_compare:nNnT { \l__enumext_mark_sym_sep_dim } = { \c_zero_dim }
3214        {
3215          \dim_set:Nn \l__enumext_mark_sym_sep_dim { \l__enumext_labelsep_v_dim }
3216        }
3217      \dim_add:Nn \l__enumext_mark_sym_sep_dim { \l__enumext_mark_sep_tmpb_dim }
3218    }
```

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

```
3219  \cs_new_protected:Nn \__enumext_keyans_show_ans:
3220    {
3221      \bool_lazy_all:nT
3222        {
3223          { \bool_if_p:N \l__enumext_show_answer_bool   }
3224          { \bool_if_p:N \l__enumext_item_wrap_key_bool }
3225        }
3226        {
3227          \__enumext_keyans_pos_mark_set:
3228          \__enumext_print_keyans_box:NN
3229            \l__enumext_labelwidth_v_dim \l__enumext_mark_sym_sep_dim
3230        }
3231    }
```

The function __enumext_keyans_show_pos: will print the ⟨*position*⟩ of the stored content in *prop list*. Need add 1 to \g__enumext_⟨*store name*⟩_prop for keyans environment.

```
3232  \cs_new_protected:Nn \__enumext_keyans_show_pos:
3233    {
3234      \int_compare:nNnTF { \l__enumext_keyans_level_int } = { 1 }
3235        {
3236          \int_incr:N \l__enumext_show_pos_tmp_int
3237        }
3238        {
3239          \int_zero:N \l__enumext_show_pos_tmp_int
```

```
3240              }
3241          \bool_lazy_all:nT
3242              {
3243                  { \bool_if_p:N \l__enumext_show_position_bool }
3244                  { \bool_if_p:N \l__enumext_item_wrap_key_bool }
3245              }
3246              {
3247                  \tl_set:Ne \l__enumext_mark_answer_sym_tl
3248                      {
3249                          \group_begin:
3250                              \exp_not:N \normalfont
3251                              \exp_not:N \footnotesize [ \int_eval:n
3252                                  {
3253                                      \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop } + \l__enumext_show
3254                                  }
3255                              ]
3256                          \group_end:
3257                      }
3258                  \__enumext_keyans_pos_mark_set:
3259                  \__enumext_print_keyans_box:NN
3260                      \l__enumext_labelwidth_v_dim \l__enumext_mark_sym_sep_dim
3261              }
3262      }
```

(*End of definition for* __enumext_keyans_pos_mark_set: , __enumext_keyans_show_ans: , *and* __enumext_keyans_show_-
pos:.)

13.34 Redefining \item and \makelabel in enumext

Redefining the \item command is not as simple as I thought. This command works in conjunction with the \makelabel command so I have to redefine both of them, in addition to this, we will have to use a couple of *global* variables to pass the values from one command to the other.

When *labeling* PDF is active \makelabel is redefined as \hss #1 and the only way to get the align key to work correctly is to redefine \makelabel using \makebox. The best way to implement this is to use the conditional command \IfDocumentMetadataTF to force this redefinition and the dedicated mode-box key to manually activate it by the user.

The \item and \item[⟨*custom*⟩] commands work in the usual way on enumext and we will add \item*, \item*[⟨*symbol*⟩] and \item*[⟨*symbol*⟩][⟨*offset*⟩].

__enumext_default_item:n   First we will see if the *optional argument* is present, if it is NOT present we will check the state of the variable \l__enumext_check_answers_bool set by the key no-store, set the boolean variable \l__enumext_-wrap_label_X_bool to "true" for the key wrap-label and execute __enumext_item_std:w and the key itemindent, otherwise we will check the state of the boolean variable \l__enumext_wrap_label_opt_-X_bool set by the key wrap-label* and execute __enumext_item_std:w with the *optional argument* and the key itemindent.

```
3263  \cs_new_protected:Npn \__enumext_default_item:n #1
3264      {
3265          \tl_if_novalue:nTF {#1}
3266              {
3267                  \bool_if:NT \l__enumext_check_answers_bool
3268                      {
3269                          \int_gincr:N \g__enumext_item_number_int
3270                          \bool_set_true:N \l__enumext_item_number_bool
3271                      }
3272                  \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
3273                  \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3274              }
3275              {
3276                  \bool_set_eq:cc
3277                      { l__enumext_wrap_label_ \__enumext_level: _bool }
3278                      { l__enumext_wrap_label_opt_ \__enumext_level: _bool }
3279                  \__enumext_item_std:w [#1] \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl
3280              }
3281      }
```

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

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

#1: `\l__enumext_item_symbol_X_tl`

#2: `\l__enumext_item_symbol_sep_X_dim`

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

```
3282 \cs_new_protected:Npn \__enumext_starred_item_exec:nn #1 #2
3283   {
3284     \tl_if_novalue:nTF {#1}
3285       {
3286         \tl_gset_eq:Nc
3287           \g__enumext_item_symbol_aux_tl { l__enumext_item_symbol_ \__enumext_level: _tl }
3288       }
3289       {
3290         \tl_gset:Nn \g__enumext_item_symbol_aux_tl {#1}
3291       }
3292     \tl_if_novalue:nTF {#2}
3293       {
3294         \dim_set_eq:cc
3295           { l__enumext_item_symbol_sep_ \__enumext_level: _dim }
3296           { l__enumext_labelsep_ \__enumext_level: _dim }
3297       }
3298       {
3299         \dim_set:cn { l__enumext_item_symbol_sep_ \__enumext_level: _dim } {#2}
3300       }
3301     \bool_if:NT \l__enumext_check_answers_bool
3302       {
3303         \int_gincr:N \g__enumext_item_number_int
3304         \bool_set_true:N \l__enumext_item_number_bool
3305       }
3306     \bool_set_true:c { l__enumext_wrap_label_ \__enumext_level: _bool }
3307     \__enumext_item_std:w \tl_use:c { l__enumext_fake_item_indent_ \__enumext_level: _tl }
3308   }
```

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

```
3309 \cs_new_protected:Nn \__enumext_starred_item_exec:
3310   {
3311     \tl_if_empty:cF { l__enumext_item_symbol_ \__enumext_level: _tl }
3312       {
3313         \mode_leave_vertical:
3314         \skip_horizontal:n { -\dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3315         \hbox_overlap_left:n { \g__enumext_item_symbol_aux_tl }
3316         \skip_horizontal:n { \dim_use:c { l__enumext_item_symbol_sep_ \__enumext_level: _dim } }
3317       }
3318   }
```

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

`\__enumext_redefine_item:` The function `\__enumext_redefine_item:` will redefine the `\item` command in the `enumext` environment adding `\item*`. This function are passed to `\__enumext_list_arg_two_X:` used in the definition of the `enumext` environment (§13.39).

```
3319 \cs_new_protected:Nn \__enumext_redefine_item:
3320   {
3321     \RenewDocumentCommand \item { s o o }
3322       {
3323         \bool_if:nTF {##1}
3324           {
3325             \__enumext_starred_item_exec:nn {##2} {##3}
3326           }
3327           { \__enumext_default_item:n {##2} }
3328       }
3329   }
```

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

__enumext_make_label:
__enumext_make_label_std:
__enumext_make_label_box:

The function `\__enumext_make_label:` redefine `\makelabel` for the keys mode-box, align, font, wrap-label, wrap-label* and `\item*` for enumext environment. This function are passed to `\__enumext_list_arg_two_X:` used in the definition of the enumext environment (§13.39).

```
3330  \cs_new_protected:Nn \__enumext_make_label:
3331    {
3332      \IfDocumentMetadataTF
3333        {
3334          \__enumext_make_label_box:
3335        }
3336        {
3337          \bool_if:NTF \l__enumext_mode_box_bool
3338            {
3339              \__enumext_make_label_box:
3340            }
3341            {
3342              \__enumext_make_label_std:
3343            }
3344        }
3345    }
```

Standard definition when `\DocumentMetadata` is not active.

```
3346  \cs_new_protected:Nn \__enumext_make_label_std:
3347    {
3348      \RenewDocumentCommand \makelabel { m }
3349        {
3350          \tl_use:c { l__enumext_label_fill_left_ \__enumext_level: _tl }
3351          \__enumext_starred_item_exec:
3352          \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3353          \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3354            {
3355              \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3356            }
3357            { ##1 }
3358          \tl_use:c { l__enumext_label_fill_right_ \__enumext_level: _tl }
3359          \tl_gclear:N \g__enumext_item_symbol_aux_tl
3360        }
3361    }
```

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

🔷 Here it is necessary to use `\strut\smash` to maintain text *alignment* in case the user wants to use `\labelbx` for example. In my experiments with *mimicking* the description environment it was the only way out and it seems to have no adverse effects and may serve in the future as a basis for a more generic list environment package than enumext.

```
3362  \cs_new_protected:Nn \__enumext_make_label_box:
3363    {
3364      \RenewDocumentCommand \makelabel { m }
3365        {
3366          \strut\smash
3367            {
3368              \makebox
3369                [ \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim } ]
3370                [ \str_use:c { l__enumext_align_label_pos_ \__enumext_level: _str } ]
3371                {
3372                  \__enumext_starred_item_exec:
3373                  \tl_use:c { l__enumext_label_font_style_ \__enumext_level: _tl }
3374                  \bool_if:cTF { l__enumext_wrap_label_ \__enumext_level: _bool }
3375                    {
3376                      \use:c { __enumext_wrapper_label_ \__enumext_level: :n } { ##1 }
3377                    }
3378                    { ##1 }
3379                  \tl_gclear:N \g__enumext_item_symbol_aux_tl
3380                }
3381            } % close smash
3382        }
3383    }
```

(*End of definition for* `\__enumext_make_label:` *,* `\__enumext_make_label_std:` *, and* `\__enumext_make_label_box:` *.*)

13.35   Setting `item-sym*` and `item-pos*` keys

In order to have a cleaner implementation of `\item*` for the enumext and enumext* environments it is best to define a couple of keys that allow us to control and set by default the ⟨*symbol*⟩ and its ⟨*offset*⟩.

item-sym*
item-pos*

Define and set `item-sym*` and `item-pos*` keys for `enumext` and `enumext*`.

```
3384 \cs_set_protected:Npn \__enumext_tmp:nn #1 #2
3385   {
3386     \keys_define:nn { enumext / #1 }
3387       {
3388         item-sym* .tl_set:c  = { l__enumext_item_symbol_#2_tl },
3389         item-sym* .value_required:n = true,
3390         item-sym* .initial:n  = {\textborn},
3391         item-pos* .dim_set:c  = { l__enumext_item_symbol_sep_#2_dim },
3392         item-pos* .value_required:n = true,
3393       }
3394   }
3395 \clist_map_inline:nn
3396   {
3397     {level-1}{i}, {level-2}{ii}, {level-3}{iii}, {level-4}{iv}, {enumext*}{vii}
3398   }
3399   { \__enumext_tmp:nn #1 }
```

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

13.36 Handling unknown keys

At this point in the code I already know that I will NOT add more ⟨*keys*⟩ for and since I have already been quite *paranoid and restrictive* with the definitions of environments and commands, the only thing left to do is do it with the ⟨*keys*⟩ (you have to be consistent in life).

💣 Well, the paragraph above is not so real, after all I had to add more ⟨*keys*⟩ than I had planned, not everything turns out the way one thinks in life.

13.36.1 Handling unknown keys for keyans, keyans* and keyanspic

unknown
__enumext_keyans_unknown_keys:n
__enumext_keyans_unknown_keys:nn

Define and set `unknown` key for `keyans`, `keyans*` and `keyanspic` environments. Here it is necessary to set `\l__enumext_envir_name_tl` in case an `unknown` key is passed using `\setenumext`.

```
3400 \cs_set_protected:Npn \__enumext_tmp:n #1
3401   {
3402     \keys_define:nn { enumext / #1 }
3403       {
3404         unknown .code:n = {
3405                           \tl_set:Nn \l__enumext_envir_name_tl {#1}
3406                           \__enumext_keyans_unknown_keys:n {##1}
3407                         },
3408       }
3409   }
3410 \clist_map_inline:nn { keyans, keyans*, keyanspic } { \__enumext_tmp:n {#1} }
```

Internal functions for handling `unknown` key.

```
3411 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:n #1
3412   {
3413     \exp_args:NV \__enumext_keyans_unknown_keys:nn \l_keys_key_str {#1}
3414   }
3415 \cs_new_protected:Npn \__enumext_keyans_unknown_keys:nn #1#2
3416   {
3417     \tl_if_blank:nTF {#2}
3418       {
3419         \msg_error:nne { enumext } { keyans-unknown-key } {#1}
3420       }
3421       {
3422         \msg_error:nnee { enumext } { keyans-unknown-key-value } {#1} {#2}
3423       }
3424   }
```

(*End of definition for* `unknown`, `\__enumext_keyans_unknown_keys:n`, *and* `\__enumext_keyans_unknown_keys:nn`.)

13.36.2 Handling unknown keys for enumext*

unknown
__enumext_starred_unknown_keys:n
__enumext_starred_unknown_keys:nn

Define and set `unknown` key for `enumext*` environment.

```
3425 \keys_define:nn { enumext / enumext* }
3426   {
3427     unknown .code:n = { \__enumext_starred_unknown_keys:n {#1} },
3428   }
```

Internal functions for handling `unknown` key.

```
3429 \cs_new_protected:Npn \__enumext_starred_unknown_keys:n #1
3430   {
3431     \exp_args:NV \__enumext_starred_unknown_keys:nn \l_keys_key_str {#1}
```

```
3432      }
3433  \cs_new_protected:Npn \__enumext_starred_unknown_keys:nn #1#2
3434    {
3435      \tl_if_blank:nTF {#2}
3436        {
3437          \msg_error:nnn { enumext } { starred-unknown-key } {#1}
3438        }
3439        {
3440          \msg_error:nnnn { enumext } { starred-unknown-key-value } {#1} {#2}
3441        }
3442    }
```

(*End of definition for* unknown *,* __enumext_starred_unknown_keys:n *, and* __enumext_starred_unknown_keys:nn*.*)

13.36.3 Handling unknown keys for enumext

unknown
__enumext_standar_unknown_keys:n
__enumext_standar_unknown_keys:nn

Defines and set the key unknown for enumext environment.

```
3443  \cs_set_protected:Npn \__enumext_tmp:n #1
3444    {
3445      \keys_define:nn { enumext / #1 }
3446        {
3447          unknown .code:n = { \__enumext_standar_unknown_keys:n {##1} },
3448        }
3449    }
3450  \clist_map_inline:nn { level-1,level-2,level-3,level-4 } { \__enumext_tmp:n {#1} }
```

Internal functions for handling unknown key.

```
3451  \cs_new_protected:Npn \__enumext_standar_unknown_keys:n #1
3452    {
3453      \exp_args:NV \__enumext_standar_unknown_keys:nn \l_keys_key_str {#1}
3454    }
3455  \cs_new_protected:Npn \__enumext_standar_unknown_keys:nn #1#2
3456    {
3457      \tl_if_blank:nTF {#2}
3458        {
3459          \msg_error:nnn { enumext } { standar-unknown-key } {#1}
3460        }
3461        {
3462          \msg_error:nnnn { enumext } { standar-unknown-key-value } {#1} {#2}
3463        }
3464    }
```

(*End of definition for* unknown *,* __enumext_standar_unknown_keys:n *, and* __enumext_standar_unknown_keys:nn*.*)

13.37 Redefining \item and \makelabel in keyans

The \item and \item[⟨*custom*⟩] commands work in the usual way in keyans, but the \item* and \item*[⟨*content*⟩] commands *store* the current ⟨*label*⟩ next to the ⟨*content*⟩ if it is present in the *sequence* and *prop list* defined by save-ans key.

__enumext_keyans_default_item:n

The function __enumext_keyans_default_item:n executes the original behavior of the \item along with the keys wrap-label, wrap-label* and itemindent.

```
3465  \cs_new_protected:Npn \__enumext_keyans_default_item:n #1
3466    {
3467      \tl_if_novalue:nTF { #1 }
3468        {
3469          \bool_set_true:N \l__enumext_wrap_label_v_bool
3470          \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
3471        }
3472        {
3473          \bool_set_eq:NN \l__enumext_wrap_label_v_bool \l__enumext_wrap_label_opt_v_bool
3474          \__enumext_item_std:w [#1] \tl_use:N \l__enumext_fake_item_indent_v_tl
3475        }
3476    }
```

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

__enumext_keyans_starred_item:n

The function __enumext_keyans_starred_item:n will take as argument #1 the *optional argument* [⟨*content*⟩] passed to \item* and save it via the __enumext_keyans_save_item_opt:n function, then activate the wrap-label key, execute \item using __enumext_item_std:w, the itemindent key and print the *optional argument* using the __enumext_keyans_show_item_opt: function handled by the wrap-opt key.

```
3477  \cs_new_protected:Npn \__enumext_keyans_starred_item:n #1
3478    {
```

```
3479     \__enumext_keyans_save_item_opt:n { #1 }
3480     \bool_set_true:N \l__enumext_wrap_label_v_bool
3481     \__enumext_item_std:w \tl_use:N \l__enumext_fake_item_indent_v_tl
3482     \__enumext_keyans_show_item_opt:
```

Now *store* the current ⟨*label*⟩ first in the *prop list* (including the *optional argument*), run the internal *"label and ref"* system if the `save-ref` key is active, then *store* in the *sequence* and finally increments `\g__enumext_-check_starred_cmd_int` for internal check system.

```
3483     \__enumext_keyans_addto_prop:n { #1 }
3484     \__enumext_keyans_store_ref:
3485     \__enumext_keyans_addto_seq:n { #1 }
3486     \int_gincr:N \g__enumext_check_starred_cmd_int
3487   }
```

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

`\item*`
`\__enumext_keyans_redefine_item:`

The function `\__enumext_keyans_redefine_item:` is responsible for adding the *starred argument* and *optional argument* by the `\__enumext_list_arg_two_v:` function in the definition of the `keyans` environment. Here we will set to true the variable `\l__enumext_item_wrap_key_bool` used by the `wrap-key` key only when `\item*` is executed and additionally we need to use `\peek_remove_spaces:n` to avoid an unwanted space when using `\item*` together with the `itemindent` key. This function are passed to `\__enumext_list_arg_two_v:` used in the definition of the `keyans` environment (§13.38).

```
3488 \cs_new_protected:Nn \__enumext_keyans_redefine_item:
3489   {
3490     \RenewDocumentCommand \item { s o }
3491       {
3492         \bool_if:nTF {##1}
3493           {
3494             \bool_set_true:N \l__enumext_item_wrap_key_bool % wrap-key
3495             \peek_remove_spaces:n
3496               {
3497                 \__enumext_keyans_starred_item:n {##2}
3498               }
3499           }
3500           {
3501             \bool_set_false:N \l__enumext_item_wrap_key_bool
3502             \__enumext_keyans_default_item:n {##2}
3503           }
3504       }
3505   }
```

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

`\__enumext_keyans_make_label:`
`\__enumext_keyans_wrapper_label:n`
`\__enumext_keyans_make_label_std:`
`\__enumext_keyans_make_label_box:`

The function `\__enumext_keyans_make_label:` redefine `\makelabel` for the keys `mode-box`, `align`, `font`, `wrap-label`, `wrap-label*`, `wrap-key` and `\item*` for `keyans` environment. This function are passed to `\__enumext_list_arg_two_v:` used in the definition of the `keyans` environment (§13.38).

```
3506 \cs_new_protected:Nn \__enumext_keyans_make_label:
3507   {
3508     \IfDocumentMetadataTF
3509       {
3510         \__enumext_keyans_make_label_box:
3511       }
3512       {
3513         \bool_if:NTF \l__enumext_mode_box_bool
3514           {
3515             \__enumext_keyans_make_label_box:
3516           }
3517           {
3518             \__enumext_keyans_make_label_std:
3519           }
3520       }
3521   }
```

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

```
3522 \cs_new_protected:Npn \__enumext_keyans_wrapper_label:n #1
3523   {
3524     \bool_lazy_all:nT
3525       {
3526         { \bool_if_p:N \l__enumext_wrap_label_v_bool      }
3527         { \bool_if_p:N \l__enumext_show_answer_bool      }
```

```
3528        { \bool_if_p:N \l__enumext_item_wrap_key_bool       }
3529        { \cs_if_exist_p:N \__enumext_keyans_wrapper_item:n }
3530      }
3531      {
3532        \cs_set_eq:NN \__enumext_wrapper_label_v:n \__enumext_keyans_wrapper_item:n
3533      }
3534    \bool_if:NTF \l__enumext_wrap_label_v_bool
3535      {
3536        \__enumext_wrapper_label_v:n { #1 }
3537      }
3538      { #1 }
3539    }
```

Standard definition when `\DocumentMetadata` is not active.

```
3540  \cs_new_protected:Nn \__enumext_keyans_make_label_std:
3541    {
3542      \RenewDocumentCommand \makelabel { m }
3543        {
3544          \tl_use:N \l__enumext_label_fill_left_v_tl
3545          \__enumext_keyans_show_ans:
3546          \__enumext_keyans_show_pos:
3547          \tl_use:N \l__enumext_label_font_style_v_tl
3548          \__enumext_keyans_wrapper_label:n { ##1 }
3549          \tl_use:N \l__enumext_label_fill_right_v_tl
3550        }
3551    }
```

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

```
3552  \cs_new_protected:Nn \__enumext_keyans_make_label_box:
3553    {
3554      \RenewDocumentCommand \makelabel { m }
3555        {
3556          \strut\smash
3557            {
3558              \makebox[ \l__enumext_labelwidth_v_dim ][ \l__enumext_align_label_pos_v_str ]
3559                {
3560                  \__enumext_keyans_show_ans:
3561                  \__enumext_keyans_show_pos:
3562                  \tl_use:N \l__enumext_label_font_style_v_tl
3563                  \__enumext_keyans_wrapper_label:n { ##1 }
3564                }
3565            }
3566        }
3567    }
```

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

13.38   Second argument of the lists

At this point of the code we have already programmed most the necessary tools to create a custom `list` environment, remember that the function `\__enumext_start_list:nn` takes two arguments, the first one we have ready, the second one we will define for all the levels of the environment `enumext` and the environment `keyans`.

13.38.1   Calculation of `\leftmargin` and `\itemindent`

Consider the figure 9 where the default margins (on the left) of a list are represented.

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

The idea is to have control over these margins so that our list does not overlap the left margin of the page. The *key* relationship is that the right edge of the `\labelsep` equals the right edge of the `\itemindent`, so that the left edge of the *label box* is at `\leftmargin`+`\itemindent` minus `\labelwidth`+`\labelsep`. Thus, the handling of the margins by the package will be as shown in the figure 10.
Where the default values will look like in the figure 11.

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

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

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

Figure 11: Default horizontal lengths in enumext.

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

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

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

```
3568 \cs_new_protected:Npn \__enumext_calc_hspace:NNNNNNN #1 #2 #3 #4 #5 #6 #7
3569   {
3570     \dim_compare:nNnT { #1 } < { \c_zero_dim }
3571       {
3572         \msg_warning:nnnV { enumext } { width-non-positive }{ labelwidth }{ #1 }
3573         \dim_set:Nn #1 { \dim_abs:n { #1 } }
3574       }
3575     \dim_compare:nNnT { #2 } < { \c_zero_dim }
3576       {
3577         \msg_warning:nnnV { enumext } { width-negative }{ labelsep }{ #2 }
3578         \dim_set:Nn #2 { \dim_abs:n { #2 } }
3579       }
```

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

```
3580     \bool_if:nF #7 { \dim_set:Nn #4 { #1 + #2} }
```

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

```
3581     \dim_compare:nNnTF { #4 } < { \c_zero_dim }
3582       {
3583         \dim_set:Nn #6 { #1 + #2 - #4}
3584         \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3585       }
3586       {
3587         \dim_compare:nNnT { #4 } = { #1 + #2 }
3588           { \dim_set:Nn #6 { \c_zero_dim } }
3589         \dim_compare:nNnT { #4 } < { #1 + #2 }
3590           { \dim_set:Nn #6 { #1 + #2 - #4} }
3591         \dim_compare:nNnT { #4 } > { #1 + #2 }
3592           {
3593             \dim_set:Nn #6 { -#1 - #2 + #4}
3594             \dim_set:Nn #6 { #6*-1}
3595           }
3596         \dim_set:Nn #5 { #1 + #2 + #3 - #6 }
3597       }
3598   }
3599 \cs_generate_variant:Nn \__enumext_calc_hspace:NNNNNNN { ccccccc }
```

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

13.38.2 Setting second argument of the lists

__enumext_list_arg_two_i:
__enumext_list_arg_two_ii:
__enumext_list_arg_two_iii:
__enumext_list_arg_two_iv:
\ enumext list arg two v:

We will "not set" \leftmargini, \leftmarginii, \leftmarginiii or \leftmarginiv, in this case, we will directly set the parameters for vertical and horizontal list spacing per level.

```
3600 \cs_set_protected:Npn \__enumext_tmp:n #1
```

```
3601    {
3602      \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3603        {
3604          \__enumext_calc_hspace:ccccccc
3605            { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3606            { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3607            { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3608            { l__enumext_leftmargin_tmp_#1_bool }
3609          \clist_map_inline:nn
3610            { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3611            { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
3612          \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3613            { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
3614          \usecounter { enumX#1 }
3615          \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
3616          \str_if_eq:nnTF {#1} { v }
3617            {
3618              \__enumext_keyans_redefine_item:
3619              \__enumext_keyans_make_label:
3620              \__enumext_keyans_ref:
3621              \__enumext_keyans_fake_item_indent:
3622              \bool_if:cT { l__enumext_show_length_#1_bool }
3623                {
3624                  \msg_term:nnnn { enumext } { list-lengths-not-nested } { v } { keyans }
3625                }
3626            }
3627            {
3628              \__enumext_redefine_item:
3629              \__enumext_make_label:
3630              \__enumext_standar_ref:
3631              \__enumext_fake_item_indent:
3632              \bool_if:cT { l__enumext_show_length_#1_bool }
3633                {
3634                  \msg_term:nnne { enumext } { list-lengths } {#1}
3635                    { \int_use:N \l__enumext_level_int }
3636                }
3637            }
3638        }
3639    }
3640  \clist_map_inline:nn { i, ii, iii, iv, v } { \__enumext_tmp:n {#1} }
```

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

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

```
3641  \cs_set_protected:Npn \__enumext_tmp:n #1
3642    {
3643      \cs_new_protected:cpn { __enumext_list_arg_two_#1: }
3644        {
3645          \bool_set_true:c { l__enumext_leftmargin_tmp_#1_bool }
3646          \dim_zero:c { l__enumext_leftmargin_tmp_#1_dim }
3647          \__enumext_calc_hspace:ccccccc
3648            { l__enumext_labelwidth_#1_dim } { l__enumext_labelsep_#1_dim }
3649            { l__enumext_listoffset_#1_dim } { l__enumext_leftmargin_tmp_#1_dim }
3650            { l__enumext_leftmargin_#1_dim } { l__enumext_itemindent_#1_dim }
3651            { l__enumext_leftmargin_tmp_#1_bool }
3652          \clist_map_inline:nn
3653            { labelsep, labelwidth, itemindent, leftmargin, rightmargin, listparindent }
3654            { \dim_set_eq:cc {####1} { l__enumext_####1_#1_dim } }
3655          \clist_map_inline:nn { topsep, parsep, partopsep, itemsep }
3656            { \skip_set_eq:cc {####1} { l__enumext_####1_#1_skip } }
3657          \skip_set_eq:Nc \parsep  { l__enumext_itemsep_#1_skip }
3658          \skip_zero:N \partopsep
3659          \usecounter { enumX#1 }
3660          \setcounter { enumX#1 } { \int_eval:n { \int_use:c { l__enumext_start_#1_int } - 1 } }
3661          \__enumext_starred_ref:
3662          \str_if_eq:nnTF {#1} { vii }
3663            {
3664              \__enumext_fake_item_indent_vii:
```

```
3665                \bool_if:cT { l__enumext_show_length_vii_bool }
3666                  { \msg_term:nnnn { enumext } { list-lengths-not-nested } { vii } { enumext* } }
3667              }
3668              {
3669                \__enumext_fake_item_indent_viii:
3670                \bool_if:cT { l__enumext_show_length_#1_bool }
3671                  { \msg_term:nnnn { enumext } { list-lengths-not-nested } { #1 } { keyans* } }
3672              }
3673          }
3674        }
3675      \clist_map_inline:nn { vii, viii } { \__enumext_tmp:n {#1} }
```

(*End of definition for* __enumext_list_arg_two_vii: *and* __enumext_list_arg_two_viii:.)

13.39 The environment enumext

__enumext_safe_exec: The __enumext_safe_exec: function first call the function __enumext_is_not_nested: which sets \g__enumext_standar_bool to *"true"* if we are NOT nested within enumext*, then call the function __enumext_internal_mini_page: to create the environment __enumext_mini_page, we will increment \l__enumext_level_int to restrict nesting of the environment, set \l__enumext_standar_bool to *"true"* and finally call the function __enumext_is_on_first_level: which sets \l__enumext_standar_-first_bool to *"true"* only if the environment is NOT nested and we are at the *"first level"*.

```
3676  \cs_new_protected:Nn \__enumext_safe_exec:
3677    {
3678      \__enumext_is_not_nested:
3679      \__enumext_internal_mini_page:
3680      \int_incr:N \l__enumext_level_int
3681      \int_compare:nNnT { \l__enumext_level_int } > { 4 }
3682        { \msg_fatal:nn { enumext } { list-too-deep } }
3683      \bool_set_true:N \l__enumext_standar_bool
3684      \bool_set_false:N \l__enumext_starred_bool
3685      \__enumext_is_on_first_level:
3686    }
```

(*End of definition for* __enumext_safe_exec:.)

__enumext_parse_keys:n   The __enumext_parse_store_keys:n function first we will clear the variable \l__enumext_series_-str used by the key series and then we check if we are at the *"first level"*, if so we process the ⟨keys⟩ and then execute the function __enumext_parse_series:n used by the key series and call the function __enumext_nested_base_line_fix: used by the key base-fix, otherwise we will pass the ⟨keys⟩ to the inner levels of the environment then we execute the function __enumext_store_active_keys:n and reprocess the ⟨keys⟩ to pass them to the *sequence* if the key save-key is not active.

```
3687  \cs_new_protected:Npn \__enumext_parse_keys:n #1
3688    {
3689      \tl_if_novalue:nF {#1}
3690        {
3691          \str_clear:N \l__enumext_series_str
3692          \int_compare:nNnTF { \l__enumext_level_int } = { 1 }
3693            {
3694              \keys_set:nn { enumext / level-1 } {#1}
3695              \__enumext_parse_series:n {#1}
3696              \__enumext_nested_base_line_fix:
3697            }
3698            {
3699              \exp_args:Ne \keys_set:nn
3700                { enumext / level-\int_use:N \l__enumext_level_int } {#1}
3701            }
3702          \__enumext_store_active_keys:n {#1}
3703        }
3704    }
```

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

__enumext_start_store_level: The __enumext_start_store_level: function activate the *"storing structure"* mechanism in the *sequence* for the command \anskey and the environment anskey*.

```
3705  \cs_new_protected:Nn \__enumext_start_store_level:
3706    {
3707      \bool_lazy_all:nT
3708        {
3709          { \bool_if_p:N \l__enumext_store_active_bool }
3710          { \bool_not_p:n { \l__enumext_keyans_env_bool } }
```

```
3711        { \bool_if_p:N \g__enumext_standar_bool }
3712      }
3713      {
3714        \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3715          {
3716            \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3717            \__enumext_store_level_open:
3718          }
3719      }
```

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

```
3720      \bool_lazy_all:nT
3721        {
3722          { \bool_if_p:N \l__enumext_store_active_bool }
3723          { \bool_not_p:n { \l__enumext_keyans_env_bool } }
3724          { \int_compare_p:nNn { \l__enumext_level_h_int } = { 1 } }
3725        }
3726        {
3727          \int_compare:nNnT { \l__enumext_level_int } > { 0 }
3728            {
3729              \bool_set_true:c { l__enumext_store_upper_level_ \__enumext_level: _bool }
3730              \__enumext_store_level_open:
3731            }
3732        }
3733    }
```

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

__enumext_stop_store_level: The __enumext_stop_store_level: function stop the *"storing structure"* mechanism in the *sequence* for the command \anskey and the environment anskey*.

```
3734  \cs_new_protected:Nn \__enumext_stop_store_level:
3735    {
3736      \bool_if:cT { l__enumext_store_upper_level_ \__enumext_level: _bool }
3737        {
3738          \__enumext_store_level_close:
3739        }
3740    }
```

(*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.

```
3741  \cs_new_protected:Nn \__enumext_multicols_start:
3742    {
3743      \int_compare:nNnT
3744        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3745        {
3746          \dim_compare:nNnT
3747            { \dim_use:c { l__enumext_columns_sep_ \__enumext_level: _dim } } = { \c_zero_dim }
3748            {
3749              \dim_set:cn { l__enumext_columns_sep_ \__enumext_level: _dim }
3750                {
3751                  ( \dim_use:c { l__enumext_labelwidth_ \__enumext_level: _dim }
3752                    + \dim_use:c { l__enumext_labelsep_ \__enumext_level: _dim }
3753                  ) / \int_use:c { l__enumext_columns_ \__enumext_level: _int }
3754                  - \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3755                }
3756            }
3757          \dim_set_eq:Nc \columnsep { l__enumext_columns_sep_ \__enumext_level: _dim  }
3758          \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3759            {
3760              \dim_zero:N \columnseprule
3761            }
```

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.

```
3762          \bool_if:cF { l__enumext_minipage_active_ \__enumext_level: _bool }
3763            {
3764              \skip_zero:N \multicolsep
3765              \__enumext_multi_addvspace:
3766            }
```

```
3767          \raggedcolumns
3768          \begin{multicols}{ \int_use:c { l__enumext_columns_ \__enumext_level: _int } }
3769        }
3770    }
```

(*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:.

```
3771 \cs_new_protected:Nn \__enumext_multicols_stop:
3772    {
3773      \int_compare:nNnTF
3774        { \int_use:c { l__enumext_columns_ \__enumext_level: _int } } > { 1 }
3775        {
3776          \__enumext_stop_list:
3777          \__enumext_stop_store_level:
3778          \end{multicols}
3779          \__enumext_unskip_unkern:
3780          \__enumext_unskip_unkern:
3781          \par\addvspace{ \skip_use:c { l__enumext_multicols_below_ \__enumext_level: _skip } }
3782        }
3783        {
3784          \__enumext_stop_list:
3785          \__enumext_stop_store_level:
3786        }
3787    }
```

(*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.

```
3788 \cs_new_protected:Nn \__enumext_before_list:
3789    {
3790      \__enumext_vspace_above:
3791      \__enumext_before_args_exec:
3792      \__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.

```
3793      \dim_compare:nNnT
3794        { \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim } } > { \c_zero_dim }
3795        {
3796          \dim_set:cn { l__enumext_minipage_left_ \__enumext_level: _dim }
3797            {
3798              \linewidth
3799              - \dim_use:c { l__enumext_minipage_right_ \__enumext_level: _dim }
3800              - \dim_use:c { l__enumext_minipage_hsep_ \__enumext_level: _dim }
3801            }
```

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.

```
3802          \bool_set_true:c { l__enumext_minipage_active_ \__enumext_level: _bool }
3803          \int_gincr:N \g__enumext_minipage_stat_int
3804          \__enumext_minipage_add_space:
3805          \noindent
3806          \__enumext_mini_page{ \dim_use:c { l__enumext_minipage_left_ \__enumext_level: _dim } }
3807        }
3808      \__enumext_multicols_start:
3809    }
```

(*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.

```
3810 \cs_new_protected:Nn \__enumext_second_part:
3811   {
3812     \bool_if:cTF { l__enumext_minipage_active_ \__enumext_level: _bool }
3813       {
3814         \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3815           {
3816             \msg_warning:nn { enumext } { missing-miniright }
3817             \miniright
3818           }
3819         \int_gzero:N \g__enumext_minipage_stat_int
3820         \__enumext_unskip_unkern: % remove topsep + [partopsep]
3821         \end__enumext_mini_page
3822       }
3823       {
3824         \__enumext_multicols_stop:
3825       }
```

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.

```
3826     \__enumext_after_stop_list:
3827     \__enumext_check_ans_key_hook:
3828     \__enumext_vspace_below:
3829     \bool_set_false:N \l__enumext_standar_bool
3830     \__enumext_resume_save_counter:
3831   }
```

(*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.

```
3832 \cs_new_protected:Nn \__enumext_set_item_width:
3833   {
3834     \dim_set:Nn \itemwidth { \linewidth }
3835     \dim_compare:nT
3836       {
3837         \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim } != \c_zero_dim
3838       }
3839       {
3840         \dim_sub:Nn \itemwidth
3841           {
3842             \dim_use:c { l__enumext_listoffset_ \__enumext_level: _dim }
3843           }
3844       }
3845   }
```

(*End of definition for* __enumext_set_item_width:.)

enumext

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

```
3846 \NewDocumentEnvironment{enumext}{ O{} }
3847   {
3848     \__enumext_safe_exec:
3849     \__enumext_parse_keys:n {#1}
3850     \__enumext_before_list:
3851     \__enumext_start_store_level:
3852     \__enumext_start_list:nn
3853       { \tl_use:c { l__enumext_label_ \__enumext_level: _tl } }
3854       {
3855         \use:c { __enumext_list_arg_two_ \__enumext_level: : }
3856         \__enumext_before_keys_exec:
3857       }
3858     \__enumext_set_item_width:
3859     \__enumext_after_args_exec:
3860   }
3861   {
```

```
3862        \__enumext_second_part:
3863      }
```

(*End of definition for* enumext. *This function is documented on page 5.*)

As we don't want our check to be executed check-ans by levels but on the complete list, we will take it out of the enumext environment using the *"hook"* function __enumext_after_env:nn.

```
3864  \__enumext_after_env:nn {enumext}
3865    {
3866        \__enumext_execute_after_env:
3867    }
```

13.40   The environment keyans

The environment keyans also based on lists. The main differences with the enumext environment are the *nesting* and the way the *answers* (choice) will be stored and checked, this environment is intended exclusively for *"multiple choice questions"*.

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

```
3868  \cs_new_protected:Nn \__enumext_keyans_safe_exec:
3869    {
3870      \bool_if:NF \l__enumext_store_active_bool
3871        {
3872          \msg_error:nnnn { enumext } { wrong-place }{ keyans }{ save-ans }
3873        }
3874      \int_incr:N \l__enumext_keyans_level_int
3875      \bool_set_true:N \l__enumext_keyans_env_bool
3876      \__enumext_keyans_name_and_start:
3877      % Set false for interfering with enumext nested in keyans (yes, its possible and crayze)
3878      \bool_set_false:N \l__enumext_store_active_bool
3879      \int_compare:nNnT { \l__enumext_keyans_level_int } > { 1 }
3880        {
3881          \msg_error:nn { enumext } { keyans-nested }
3882        }
3883      \int_compare:nNnT { \l__enumext_level_int } > { 1 }
3884        {
3885          \msg_error:nn { enumext } { keyans-wrong-level }
3886        }
3887    }
```

(*End of definition for* __enumext_keyans_safe_exec:.)

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

```
3888  \cs_new_protected:Npn \__enumext_keyans_parse_keys:n #1
3889    {
3890      \keys_set:nn { enumext / keyans } {#1}
3891    }
```

(*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:

```
3892  \cs_new_protected:Nn \__enumext_before_list_v:
3893    {
3894      \__enumext_vspace_above_v:
3895      \__enumext_before_args_exec_v:
3896      \dim_compare:nNnT { \l__enumext_minipage_right_v_dim } > { \c_zero_dim }
3897        {
3898          \dim_set:Nn \l__enumext_minipage_left_v_dim
3899            {
3900              \linewidth - \l__enumext_minipage_right_v_dim - \l__enumext_minipage_hsep_v_dim
3901            }
3902          \bool_set_true:N \l__enumext_minipage_active_v_bool
3903          \int_gincr:N \g__enumext_minipage_stat_int
3904          \__enumext_keyans_minipage_add_space:
3905          \__enumext_mini_page{ \l__enumext_minipage_left_v_dim }
3906        }
3907      \__enumext_keyans_multicols_start:
3908    }
3909  \cs_new_protected:Nn \__enumext_keyans_multicols_start:
3910    {
```

```
3911        \int_compare:nNnT { \l__enumext_columns_v_int } > { 1 }
3912          {
3913            \dim_compare:nNnT { \l__enumext_columns_sep_v_dim } = { \c_zero_dim }
3914              {
3915                \dim_set:Nn \l__enumext_columns_sep_v_dim
3916                  {
3917                    (
3918                      \l__enumext_labelwidth_v_dim + \l__enumext_labelsep_v_dim
3919                    ) / \l__enumext_columns_v_int
3920                  - \l__enumext_listoffset_v_dim
3921                  }
3922              }
3923            \dim_set_eq:NN \columnsep \l__enumext_columns_sep_v_dim
3924            \dim_zero:N \columnseprule % no rule here
3925            \bool_if:NF \l__enumext_minipage_active_v_bool
3926              {
3927                \skip_zero:N \multicolsep
3928                \__enumext_keyans_multi_addvspace:
3929              }
3930            \raggedcolumns
3931            \begin{multicols}{ \l__enumext_columns_v_int }
3932          }
3933      }
3934  \cs_new_protected:Nn \__enumext_keyans_multicols_stop:
3935    {
3936      \int_compare:nNnTF { \l__enumext_columns_v_int } > { 1 }
3937        {
3938          \__enumext_stop_list:
3939          \end{multicols}
3940          \__enumext_unskip_unkern:
3941          \__enumext_unskip_unkern:
3942          \par\addvspace{ \l__enumext_multicols_below_v_skip }
3943        }
3944        {
3945          \__enumext_stop_list:
3946        }
3947      }
3948  \cs_new_protected:Nn \__enumext_second_part_v:
3949    {
3950      \bool_if:NTF \l__enumext_minipage_active_v_bool
3951        {
3952          \int_compare:nNnT { \g__enumext_minipage_stat_int } = { 1 }
3953            {
3954              \msg_warning:nn { enumext } { missing-miniright }
3955              \miniright
3956            }
3957          \int_gzero:N \g__enumext_minipage_stat_int
3958          \__enumext_unskip_unkern: % remove \topsep + [\partopsep]
3959          \end__enumext_mini_page
3960          \par\addvspace{ \l__enumext_minipage_after_skip }
3961        }
3962        {
3963          \__enumext_keyans_multicols_stop:
3964        }
3965      \bool_set_false:N \l__enumext_keyans_env_bool
3966      \__enumext_after_stop_list_v:
3967      \__enumext_vspace_below_v:
3968    }
```

(*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.

```
3969  \cs_new_protected:Nn \__enumext_keyans_set_item_width:
3970    {
3971      \dim_set:Nn \itemwidth { \linewidth }
3972      \dim_compare:nT
3973        {
3974          \l__enumext_listoffset_v_dim != \c_zero_dim
3975        }
3976        {
```

```
3977            \dim_sub:Nn \itemwidth { \l__enumext_listoffset_v_dim }
3978          }
3979      }
```

(*End of definition for* __enumext_keyans_set_item_width:.)

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

```
3980  \NewDocumentEnvironment{keyans}{ O{} }
3981    {
3982      \__enumext_keyans_safe_exec:
3983      \__enumext_keyans_parse_keys:n {#1}
3984      \__enumext_before_list_v:
3985      \__enumext_start_list:nn
3986        { \tl_use:N \l__enumext_label_v_tl }
3987        {
3988          \__enumext_list_arg_two_v:
3989          \__enumext_before_keys_exec_v:
3990        }
3991      \__enumext_keyans_set_item_width:
3992      \__enumext_after_args_exec_v:
3993    }
3994    {
3995      \__enumext_check_starred_cmd:n { item }
3996      \__enumext_second_part_v:
3997    }
```

(*End of definition for* keyans. *This function is documented on page* 15.)

13.41 Tagging PDF support for non-standart list environments

The LaTeX release 2022-06-01 brings automatic support for *tagged* PDF in several aspects, including the standard *list environments* and the list environment. Unfortunately non-standard *list environments* like keyanspic or the horizontal list environments enumext* and keyans* are not structured in a nice way, i.e. the expected result in the PDF file is the expected one, but the underlying structure is not correct. In simple terms, for *tagged* PDF a list environment is a list environment, no matter what it looks like in the PDF file.

To maintain a correct list structure when \DocumentMetadata is active, it is necessary to do some things manually using tagpdf[17] and ltsockets[19]. This implementation is an adaptation of my answer thanks to Ulrike Fischer's comments in How can I modify my \item redefinition to be compatible with tagging-pdf.

13.41.1 Socket for tagging support in enumext* and keyans*

start-list-tags
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*.

```
3998  \socket_new:nn {tagsupport/__enumext/starred}{ 1 }
3999  \socket_new_plug:nnn {tagsupport/__enumext/starred} {start-list-tags}
4000    {
4001      \tag_resume:n {#1}
4002      \tag_mc_end_push:
4003        \tag_struct_begin:n {tag=LI}
4004          \tag_struct_begin:n {tag=Lbl}
4005            \tag_mc_begin:n {tag=Lbl}
4006      }
4007  \socket_new_plug:nnn {tagsupport/__enumext/starred} {stop-start-tags}
4008    {
4009            \tag_mc_end:
4010          \tag_struct_end:n {tag=Lbl}
4011          \tag_struct_begin:n {tag=LBody}
4012            \tag_struct_begin:n {tag=text-unit}
4013              \tag_struct_begin:n {tag=text}
4014      }
4015  \socket_new_plug:nnn {tagsupport/__enumext/starred} {stop-list-tags}
4016    {
4017            \tag_struct_end:n {tag=text}
4018          \tag_struct_end:n {tag=text-unit}
4019          \tag_struct_end:n {tag=LBody}
4020        \tag_struct_end:n {tag=LI}
4021      \tag_mc_begin_pop:n {}
4022      \tag_suspend:n {#1}
4023    }
```

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

```
4024  \cs_new_protected_nopar:Npn \__enumext_start_list_tag:n #1
4025    {
```

```
4026     \IfDocumentMetadataTF
4027        {
4028           \socket_assign_plug:nn {tagsupport/__enumext/starred} {start-list-tags}
4029           \socket_use:nn {tagsupport/__enumext/starred} {#1}
4030        } {}
4031     }
4032  \cs_new_protected_nopar:Nn \__enumext_stop_start_list_tag:
4033     {
4034        \IfDocumentMetadataTF
4035           {
4036              \socket_assign_plug:nn {tagsupport/__enumext/starred} {stop-start-tags}
4037              \socket_use:nn {tagsupport/__enumext/starred} { }
4038           } {}
4039     }
4040  \cs_new_protected_nopar:Npn \__enumext_stop_list_tag:n #1
4041     {
4042        \IfDocumentMetadataTF
4043           {
4044              \socket_assign_plug:nn {tagsupport/__enumext/starred} {stop-list-tags}
4045              \socket_use:nn {tagsupport/__enumext/starred} {#1}
4046           } {}
4047     }
```

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

13.41.2   Socket for tagging support in keyanspic

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

```
4048  \socket_new:nn {tagsupport/__enumext/keyanspic}{ 0 }
4049  \socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {start-list-tags}
4050     {
4051        \tag_resume:n {keyanspic}
4052        \tag_mc_end_push:
4053          \tag_struct_begin:n {tag=LI}
4054             \tag_struct_begin:n {tag=Lbl}
4055                \tag_mc_begin:n {tag=Lbl}
4056     }
4057  \socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {stop-start-tags}
4058     {
4059                \tag_mc_end:
4060             \tag_struct_end:n {tag=Lbl}
4061             \tag_struct_begin:n {tag=LBody}
4062                \tag_struct_begin:n {tag=text-unit}
4063                   \tag_struct_begin:n {tag=text}
4064                      \tag_mc_begin:n {tag=text}
4065     }
4066  \socket_new_plug:nnn {tagsupport/__enumext/keyanspic} {stop-list-tags}
4067     {
4068                \tag_mc_end:
4069                   \tag_struct_end:n {tag=text}
4070                \tag_struct_end:n {tag=text-unit}
4071             \tag_struct_end:n {tag=LBody}
4072          \tag_struct_end:n {tag=LI}
4073        \tag_mc_begin_pop:n {}
4074        \tag_suspend:n {keyanspic}
4075     }
```

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

```
4076  \cs_new_protected_nopar:Nn \__enumext_anspic_start_list_tag:
4077     {
4078        \IfDocumentMetadataTF
4079           {
4080              \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {start-list-tags}
4081              \socket_use:n {tagsupport/__enumext/keyanspic}
4082           } {}
4083     }
4084  \cs_new_protected_nopar:Nn \__enumext_anspic_stop_start_list_tag:
4085     {
4086        \IfDocumentMetadataTF
4087           {
4088              \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {stop-start-tags}
4089              \socket_use:n {tagsupport/__enumext/keyanspic}
4090           } {}
```

```
4091      }
4092  \cs_new_protected_nopar:Nn \__enumext_anspic_stop_list_tag:
4093    {
4094      \IfDocumentMetadataTF
4095        {
4096          \socket_assign_plug:nn {tagsupport/__enumext/keyanspic} {stop-list-tags}
4097          \socket_use:n {tagsupport/__enumext/keyanspic}
4098        } {}
4099      }
```

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

13.42 The environment keyanspic and \anspic

The keyanspic environment is a list based environment that uses the same configuration for *"spacing"* and ⟨*label*⟩ as the keyans environment, but it does not use \item. The ⟨*contents*⟩ are passed to the environment by means of the \anspic command as replacement for \item command and placed inside minipage environments, with the ⟨*label*⟩ centered *"above"* or *"below"*, adjusting *widths* and *position* according to the options passed to the environment.

Figure 12: Representation of the keyanspic spacing in enumext.

In order for the keyanspic environment and the \anspic command to work correctly, we need to set and export some variables in the first part of the environment definition and pass them to \anspic which is executed in the second part of the environment. This implementation is adapted from the answer given by Enrico Gregorio (@egreg) in How to process the body of an environment and divide it by a \macro?.

13.42.1 The environment keyanspic

label-pos
label-sep
layout-sty
layout-sep
layout-top
mark-ans
mark-pos
mark-sep
save-sep
wrap-opt
wrap-key
show-ans
show-pos

First we define the key that allows us to process the position of the ⟨*label*⟩ centered "above" or "below" which will be label-pos, the vertical separation of these from *drawing or tabular* will be handled with the key label-sep. The *"layout style"* will be handled with the key layout-sty will take two values separated by comma {⟨*n° upper, n° lower*⟩} and will determine the number of minipage environments in which all arguments of \anspic will be printed at the "upper" and "lower" within the environments separated by the value of the key layout-sep. The vertical space "top" and "bottom" of the environment will be handled with the key layout-top.

```
4100  \keys_define:nn { enumext / keyanspic }
4101    {
4102      label-pos .choice:,
4103      label-pos / above   .code:n =
4104                          \bool_set_true:N \l__enumext_anspic_label_above_bool
4105                          \str_set:Nn \l__enumext_anspic_mini_pos_str { t },
4106      label-pos / below   .code:n =
4107                          \bool_set_false:N \l__enumext_anspic_label_above_bool
4108                          \str_set:Nn \l__enumext_anspic_mini_pos_str { b },
4109      label-pos / unknown .code:n =
4110                          \msg_error:nneee { enumext } { unknown-choice }
4111                            { label-pos } { above,~ below } { \exp_not:n {#1} },
4112      label-pos .initial:n     = below,
4113      label-pos .value_required:n = true,
4114      label-sep .skip_set:N     = \l__enumext_anspic_label_sep_skip,
4115      label-sep .value_required:n = true,
4116      layout-sty .tl_set:N      = \l__enumext_anspic_layout_style_tl,
4117      layout-sty .value_required:n = true,
4118      layout-sep .code:n        = \keys_set:nn { enumext / keyans } { parsep = #1 },
4119      layout-sep .value_required:n = true,
4120      layout-top .code:n        = \keys_set:nn { enumext / keyans } { topsep = #1 },
4121      layout-top .value_required:n = true,
4122      mark-ans   .code:n        = \keys_set:nn { enumext / keyans } { mark-ans = #1 },
4123      mark-ans   .value_required:n = true,
4124      mark-pos   .code:n        = \keys_set:nn { enumext / keyans } { mark-pos = #1 },
4125      mark-pos   .value_required:n = true,
```

```
4126        mark-sep    .code:n            = \keys_set:nn { enumext / keyans } { mark-sep = #1 },
4127        mark-sep    .value_required:n = true,
4128        save-sep    .code:n            = \keys_set:nn { enumext / keyans } { save-sep = #1 },
4129        save-sep    .value_required:n = true,
4130        wrap-opt    .code:n            = \keys_set:nn { enumext / keyans } { wrap-opt = #1 },
4131        wrap-opt    .value_required:n = true,
4132        wrap-key    .code:n            = \keys_set:nn { enumext / keyans } { wrap-key = #1 },
4133        wrap-key    .value_required:n = true,
4134        show-ans    .code:n            = \keys_set:nn { enumext / keyans } { show-ans = #1 },
4135        show-ans    .value_required:n = true,
4136        show-pos    .code:n            = \keys_set:nn { enumext / keyans } { show-pos = #1 },
4137        show-pos    .value_required:n = true,
4138        unknown     .code:n            = {
4139                                          \tl_set:Nn \l__enumext_envir_name_tl { keyanspic }
4140                                          \__enumext_keyans_unknown_keys:n {#1}
4141                                        },
4142      }
```

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

__enumext_keyans_pic_safe_exec:
__enumext_keyans_pic_parse_keys:n
__enumext_keyans_pic_skip_abs:N
__enumext_keyans_pic_arg_two:

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

```
4143 \cs_new_protected:Nn \__enumext_keyans_pic_safe_exec:
4144   {
4145     \int_incr:N \l__enumext_keyans_pic_level_int
4146     \int_compare:nNnT { \l__enumext_keyans_pic_level_int } > { 1 }
4147       {
4148         \msg_error:nn { enumext } { keyanspic-nested }
4149       }
4150     \__enumext_keyans_name_and_start:
4151   }
```

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

```
4152 \cs_new_protected:Npn \__enumext_keyans_pic_parse_keys:n #1
4153   {
4154     \tl_if_novalue:nF {#1}
4155       {
4156         \keys_set:nn { enumext / keyanspic } {#1}
4157       }
4158   }
```

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

```
4159 \cs_new_protected:Npn \__enumext_keyans_pic_skip_abs:N #1
4160   {
4161     \dim_compare:nNnT { #1 } < { \c_zero_dim }
4162       {
4163         \skip_set:Nn #1 { -#1 }
4164       }
4165   }
```

The __enumext_keyans_pic_arg_two: function will be used in the *second argument* of the list environment that defines the keyanspic environment, with this we will take the configuration of the *"spaces"* and the keys label, wrap-label, parsep and topsep from the keyans environment. The first thing we need to do is set the boolean variable \l__enumext_leftmargin_tmp_v_bool handled by the list-indent key to "false", then copy the definition of the second list argument from the keyans environment definition and make sure that \parsep does not have a negative value.

```
4166 \cs_new_protected:Npn \__enumext_keyans_pic_arg_two:
4167   {
4168     \bool_set_false:N \l__enumext_leftmargin_tmp_v_bool
4169     \__enumext_list_arg_two_v:
4170     \__enumext_keyans_pic_skip_abs:N \parsep
```

Now we increment the counter enumXv of the keyans environment and save the *total height* of the ⟨*label*⟩ in \l__enumext_anspic_label_htdp_dim used by \anspic and we will adjust the values of \parsep only if the key label-pos is set to *below*.

```
4171     \bool_if:NF \l__enumext_anspic_label_above_bool
4172       {
4173         \stepcounter { enumXv }
4174         \hbox_set:Nn \l__enumext_anspic_label_box { \l__enumext_label_v_tl }
4175         \dim_set:Nn \l__enumext_anspic_label_htdp_dim
4176           {
```

```
4177          \box_ht_plus_dp:N \l__enumext_anspic_label_box
4178        }
4179      \skip_add:Nn \parsep
4180        {
4181          \l__enumext_anspic_label_htdp_dim
4182          + \box_dp:N \strutbox
4183          + \l__enumext_anspic_label_sep_skip
4184        }
4185    }
```

Finally we *adjust* the value of \leftmargin and \topsep then set \listparindent, \partopsep and \itemsep to zero so that the *horizontal* and *vertical* space is not affected.

```
4186    \dim_add:Nn \leftmargin { -\l__enumext_labelwidth_v_dim - \l__enumext_labelsep_v_dim }
4187    \ignorespaces
4188    \skip_add:Nn \topsep { 0.5\box_dp:N \strutbox }
4189    \dim_zero:N  \listparindent
4190    \skip_zero:N \partopsep
4191    \skip_zero:N \itemsep
4192  }
```

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

keyanspic   Now we define the environment keyanspic. For compatibility with *tagged* PDF we must use the \begin{list} form and a lot of conditional code using \IfDocumentMetadataTF. We will first stop the code for automatic *tagged* PDF for list environments, redefine \item so that it cannot be used, and stop the code for automatic *tagged* PDF for the keyanspic environment.

```
4193  \NewDocumentEnvironment{keyanspic}{ o }
4194    {
4195      \__enumext_keyans_pic_safe_exec:
4196      \__enumext_keyans_pic_parse_keys:n {#1}
4197      \begin{list} { } { \__enumext_keyans_pic_arg_two: }
4198      \IfDocumentMetadataTF
4199        {
4200          \tag_suspend:n {list}
4201        }{}
4202      \item[] \scan_stop:
4203      \RenewDocumentCommand \item {}
4204        {
4205          \msg_error:nn { enumext } { keyanspic-item-cmd }
4206        }
4207      \IfDocumentMetadataTF
4208        {
4209          \tag_resume:n {keyanspic}
4210          \tag_tool:n {para/tagging=false}
4211          \tag_suspend:n {keyanspic}
4212        } { }
4213    }
```

In the second part of the environment definition we will manually place our code for *tagged* PDF and execute the command \anspic using the __enumext_anspic_exec: function.

```
4214    {
4215      \IfDocumentMetadataTF
4216        {
4217          \tag_resume:n {keyanspic}
4218          \tag_mc_end_push:
4219          \tag_struct_begin:n {tag=L,attribute=enumerate}
4220        } { }
4221      \__enumext_anspic_exec:
4222      \IfDocumentMetadataTF
4223        {
4224          \tag_suspend:n {keyanspic}
4225        } { }
4226      \end{list}
4227      \IfDocumentMetadataTF
4228        {
4229          \tag_struct_end:n {tag=L}
4230          \tag_mc_begin_pop:n {}
4231          \tag_struct_end:n {tag=L}
4232          \tag_mc_begin_pop:n {}
4233        } { }
```

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

```
4234        \__enumext_check_starred_cmd:n { anspic }
4235        \setcounter { enumXvi } { 0 }
4236        \bool_if:NTF \l__enumext_anspic_label_above_bool
4237          {
4238            \par\addvspace{ 0.5\box_dp:N \strutbox }
4239          }
4240          {
4241            \par
4242            \addvspace
4243              {
4244                \dim_eval:n
4245                  {
4246                    \l__enumext_anspic_label_htdp_dim + \box_ht_plus_dp:N \strutbox
4247                    + \l__enumext_anspic_label_sep_skip + \l__enumext_topsep_v_skip
4248                  }
4249              }
4250          }
4251      }
```

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

13.42.2 The command \anspic

The `\anspic` command take three arguments, the *starred versions* `\anspic*`[⟨*content*⟩] *store* the current ⟨*label*⟩ next to the *optional argument* [⟨*content*⟩] in the *sequence* and *prop list* defined by save-ans key. The third *mandatory argument* {⟨*drawing or tabular*⟩} is NOT *stored* in the *sequence* or *prop list*.

🔵 One of the complications here to make the keyanspic environment compatible with *tagged* PDF is the position of ⟨*label*⟩, the `\anspic` command processes the arguments in order, where #1 and #2 correspond to ⟨*label*⟩ and #3 to the mandatory argument and puts all this inside a minipage environment. If #1 and #2, that is ⟨*label*⟩, is above #3 there are no problems with *tagged* PDF, but if #3 comes first the list created with *tagged* PDF will not be correct.

\anspic
__enumext_anspic_body_dim:n
__enumext_anspic_label:nn
__enumext_anspic_label_pos:nnn
__enumext_anspic_args:nnn
__enumext_anspic_print:n
__enumext_anspic_print:e
__enumext_anspic_print:V
__enumext_anspic_row:n
__enumext_anspic_exec:

We check that the command is active in the keyanspic environment only if the save-ans key is present, otherwise we return an error. The three arguments are handled by the function `\__enumext_anspic_args:nnn` and stored in the sequence `\l__enumext_anspic_args_seq` which is processed by the keyanspic environment.

```
4252  \NewDocumentCommand \anspic { s o +m }
4253    {
4254      \bool_if:NF \l__enumext_store_active_bool
4255        {
4256          \msg_error:nnnn { enumext } { wrong-place }{ keyanspic }{ save-ans }
4257        }
4258      \int_compare:nNnT { \l__enumext_level_int } > { 1 }
4259        {
4260          \msg_error:nn { enumext } { keyanspic-wrong-level }
4261        }
4262      \int_compare:nNnT { \l__enumext_keyans_level_int } = { 1 }
4263        {
4264          \msg_error:nnnn { enumext } { command-wrong-place }{ anspic }{ keyans }
4265        }
4266      \seq_put_right:Nn \l__enumext_anspic_args_seq
4267        {
4268          \__enumext_anspic_args:nnn { #1 } { #2 } { #3 }
4269        }
4270    }
```

The `\__enumext_anspic_body_dim:n` function will set the value of `\l__enumext_anspic_body_htdp_-dim` equal to the "height plus depth" of the *mandatory argument* if the key label-pos is set "below".

```
4271  \cs_new_protected:Npn \__enumext_anspic_body_dim:n #1
4272    {
4273      \bool_if:NF \l__enumext_anspic_label_above_bool
4274        {
4275          \IfDocumentMetadataTF
4276            {
4277              \tag_suspend:n {keyanspic}
4278            } { }
4279          \vbox_set:Nn \l__enumext_anspic_body_box { #1 }
4280          \dim_set:Nn \l__enumext_anspic_body_htdp_dim
4281            {
4282              \box_ht_plus_dp:N \l__enumext_anspic_body_box
```

```
4283                    }
4284                \IfDocumentMetadataTF
4285                    {
4286                        \tag_resume:n {keyanspic}
4287                    } { }
4288                }
4289            }
```

The `\__enumext_anspic_label:nn` function will process inside `\makebox` the *starred argument* '`*`' and *optional argument* passed to the command. Here we will store the ⟨*label*⟩ and *optional argument* in *prop list* and *sequence* and execute the `show-ans`, `show-pos`, `font`, `wrap-label`, `wrap-key` and `wrap-opt` keys.

```
4290    \cs_new_protected:Npn \__enumext_anspic_label:nn #1 #2
4291        {
4292            \makebox[ \l__enumext_anspic_mini_width_dim ][ c ]
4293                {
4294                    \bool_if:nTF { #1 }
4295                        {
4296                            \bool_set_true:N \l__enumext_item_wrap_key_bool
4297                            \bool_set_true:N \l__enumext_wrap_label_v_bool
4298                            \__enumext_keyans_save_item_opt:n { #2 }
4299                            \__enumext_keyans_addto_prop:n { #2 }
4300                            \__enumext_keyans_store_ref:
4301                            \__enumext_keyans_addto_seq:n { #2 }
4302                            \int_gincr:N \g__enumext_check_starred_cmd_int
4303                            \__enumext_keyans_show_ans:
4304                            \__enumext_keyans_show_pos:
4305                            \makebox[ \l__enumext_labelwidth_v_dim ][c]
4306                                {
4307                                    \tl_use:N \l__enumext_label_font_style_v_tl
4308                                    \__enumext_keyans_wrapper_label:n { \l__enumext_label_vi_tl }
4309                                }
4310                            \skip_horizontal:n { \l__enumext_labelsep_v_dim }
4311                            \__enumext_keyans_show_item_opt:
4312                        }
4313                        {
4314                            \bool_set_false:N \l__enumext_item_wrap_key_bool
4315                            \tl_use:N \l__enumext_label_font_style_v_tl
4316                            \__enumext_wrapper_label_v:n { \l__enumext_label_vi_tl }
4317                        }
4318                }
4319        }
```

The function `\__enumext_anspic_label_pos:nnn` will be in charge of handling the *"counter"* and the position of the ⟨*label*⟩, set by `label-pos` key which will have the same configuration as the `keyans` environment.

```
4320    \cs_new_protected:Npn \__enumext_anspic_label_pos:nnn #1 #2 #3
4321        {
4322            \stepcounter { enumXvi }
4323            \__enumext_anspic_body_dim:n { #3 }
4324            \bool_if:NTF \l__enumext_anspic_label_above_bool
4325                {
4326                    \__enumext_anspic_label:nn { #1 } { #2 }
4327                }
4328                {
4329                    \raisebox
4330                        {
4331                            -\dim_eval:n
4332                                {
4333                                    \l__enumext_anspic_label_htdp_dim
4334                                    + \l__enumext_anspic_body_htdp_dim
4335                                    + \box_dp:N \strutbox
4336                                    + \l__enumext_anspic_label_sep_skip
4337                                }
4338                        }
4339                        [ 0pt ] [ 0pt ]
4340                        {
4341                            \__enumext_anspic_label:nn { #1 } { #2 }
4342                        }
4343                }
4344        }
4345 %
```

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.

```
4346  \cs_new_protected:Nn \__enumext_anspic_args:nnn
4347    {
4348      \__enumext_anspic_start_list_tag:
4349      \__enumext_anspic_label_pos:nnn { #1 } { #2 } { #3 }
4350      \__enumext_anspic_stop_start_list_tag:
4351      \bool_if:NTF \l__enumext_anspic_label_above_bool
4352        {
4353          \\[\l__enumext_anspic_label_sep_skip] #3
4354        }
4355        {
4356          \\ #3
4357        }
4358      \__enumext_anspic_stop_list_tag:
4359    }
```

The value {⟨*n° upper, n° lower*⟩} passed to the layout-sty key is split by comma and is handled directly by the function `\__enumext_anspic_print:n` and passed to the function `\__enumext_anspic_row:n`.

```
4360  \cs_new_protected:Nn \__enumext_anspic_print:n
4361    {
4362      \clist_map_function:nN { #1 } \__enumext_anspic_row:n
4363    }
4364  \cs_generate_variant:Nn \__enumext_anspic_print:n { e, V }
```

The function `\__enumext_anspic_row:n` will set the *widths* for the minipage environments and place *all arguments* passed to `\anspic` *saved* in the `\l__enumext_anspic_args_seq` sequence inside them.

```
4365  \cs_new_protected:Nn \__enumext_anspic_row:n
4366    {
4367      \dim_set:Nn \l__enumext_anspic_mini_width_dim { \linewidth / #1 }
4368      \int_set:Nn \l__enumext_anspic_above_int { \l__enumext_anspic_below_int }
4369      \int_set:Nn \l__enumext_anspic_below_int { \l__enumext_anspic_above_int + #1 }
4370      \int_step_inline:nnn
4371        { \l__enumext_anspic_above_int + 1 }
4372        { \l__enumext_anspic_below_int }
4373        {
4374          \IfDocumentMetadataTF
4375            {
4376              \tag_suspend:n {minipage}
4377            } { }
4378          \begin{minipage}[ \l__enumext_anspic_mini_pos_str ]{ \l__enumext_anspic_mini_width_dim }
4379            \centering
4380            \seq_item:Nn \l__enumext_anspic_args_seq { ##1 }
4381          \end{minipage}
4382          \IfDocumentMetadataTF
4383            {
4384              \tag_resume:n {minipage}
4385            } { }
4386        }
4387      \par
4388    }
```

The `\__enumext_anspic_exec:` function will execute all the code in the `\anspic` command in the second argument of the keyanspic environment definition. If the key layout-sty is not set, everything will be printed on a *single line*.

```
4389  \cs_new_protected:Nn \__enumext_anspic_exec:
4390    {
4391      \tl_if_empty:NTF \l__enumext_anspic_layout_style_tl
4392        {
4393          \__enumext_anspic_print:e { \seq_count:N \l__enumext_anspic_args_seq }
4394        }
4395        {
4396          \__enumext_anspic_print:V \l__enumext_anspic_layout_style_tl
4397        }
4398    }
```

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

13.43 The horizontal environments

Generating *horizontal list environments* is NOT as simple as standard LaTeX list environments. The fundamental part of the code is adapted from the `shortlst` package to a more modern version using `expl3`. It is not possible to redefine `\item` and `\makelabel` using `\RenewDocumentCommand` as in the vertical *non starred* versions.

To achieve the *horizontal list environments* we will capture the `\item` command and the ⟨*content*⟩ of this in *horizontal box* using `\makebox` for the `label` and a `minipage` environment for the ⟨*content*⟩ passed to `\item`, we will also add the *optional argument* (⟨*number*⟩) to `\item` to be able to *join columns* horizontally, in simple terms, we want `\item` to behave in the same way as in the `enumext` environment but adding an *first optional argument* (⟨*number*⟩).

A side effect is the limitation of using `\item` in this way *without* using `\RenewDocumentCommand`, which loses the original definition and affects the *standard list environments* provided by LaTeX and any environment defined using base `list` environment, including: `itemize`, `enumerate`, `description`, `quote`, `quotation`, `verse`, `center`, `flushleft`, `flushright`, `verbatim`, `tabbing`, `trivlist`, `list` and all environments created with `\newtheorem`.

💣 One way to get around this is to use something like:

```
\AddToHook{env/enumerate/before}{recover original \item definition}
```

inside `minipage`, but in my partial tests this does not have the desired effect and the vertical and horizontal spacing is distorted. For now this will remain as a limitation and I will see if it is feasible to implement it in the future.

🔶 For compatibility with the *tagged* PDF we close the environments according to the presence or not of the `mini-env` key.

13.43.1 Functions for item box width

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:`

```
4399 \cs_new_protected:Nn \__enumext_starred_columns_set_vii:
4400   {
4401     \dim_compare:nNnT { \l__enumext_columns_sep_vii_dim } = { \c_zero_dim }
4402       {
4403         \dim_set:Nn \l__enumext_columns_sep_vii_dim
4404           {
4405             ( \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim )
4406             / \l__enumext_columns_vii_int
4407           }
4408       }
4409     \int_set:Nn \l__enumext_tmpa_vii_int { \l__enumext_columns_vii_int - 1 }
4410     \dim_set:Nn \l__enumext_item_width_vii_dim
4411       {
4412         ( \linewidth - \l__enumext_columns_sep_vii_dim * \l__enumext_tmpa_vii_int )
4413         / \l__enumext_columns_vii_int
4414         - \l__enumext_labelwidth_vii_dim
4415         - \l__enumext_labelsep_vii_dim
4416       }
```

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

```
4417     \dim_compare:nNnT { \l__enumext_rightmargin_vii_dim } > { \c_zero_dim }
4418       {
4419         \dim_sub:Nn \l__enumext_item_width_vii_dim
4420           {
4421             ( \l__enumext_rightmargin_vii_dim * \l__enumext_tmpa_vii_int )
4422             / \l__enumext_columns_vii_int
4423           }
4424         \dim_add:Nn \l__enumext_columns_sep_vii_dim
4425           {
4426             \l__enumext_rightmargin_vii_dim
4427           }
4428       }
4429   }
```

Same implementation for the `keyans*` environment.

```
4430 \cs_new_protected:Nn \__enumext_starred_columns_set_viii:
4431   {
4432     \dim_compare:nNnT { \l__enumext_columns_sep_viii_dim } = { \c_zero_dim }
4433       {
4434         \dim_set:Nn \l__enumext_columns_sep_viii_dim
4435           {
4436             ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim )
4437             / \l__enumext_columns_viii_int
4438           }
4439       }
```

```
4440        \int_set:Nn \l__enumext_tmpa_viii_int { \l__enumext_columns_viii_int - 1 }
4441        \dim_set:Nn \l__enumext_item_width_viii_dim
4442          {
4443            ( \linewidth - \l__enumext_columns_sep_viii_dim * \l__enumext_tmpa_viii_int )
4444            / \l__enumext_columns_viii_int
4445            - \l__enumext_labelwidth_viii_dim
4446            - \l__enumext_labelsep_viii_dim
4447          }
4448        \dim_compare:nNnT { \l__enumext_rightmargin_viii_dim } > { \c_zero_dim }
4449          {
4450            \dim_sub:Nn \l__enumext_item_width_viii_dim
4451              {
4452                ( \l__enumext_rightmargin_viii_dim * \l__enumext_tmpa_vii_int )
4453                / \l__enumext_columns_viii_int
4454              }
4455            \dim_add:Nn \l__enumext_columns_sep_viii_dim
4456              {
4457                \l__enumext_rightmargin_viii_dim
4458              }
4459          }
4460      }
```

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

13.43.2   Functions for join item columns

__enumext_starred_joined_item_vii:n
__enumext_starred_joined_item_viii:n

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

```
4461  \cs_new_protected:Npn \__enumext_starred_joined_item_vii:n #1
4462    {
4463      \int_set:Nn \l__enumext_joined_item_vii_int {#1}
4464      \int_compare:nNnT { \l__enumext_joined_item_vii_int } > { \l__enumext_columns_vii_int }
4465        {
4466          \msg_warning:nnee { enumext } { item-joined }
4467            { \int_use:N \l__enumext_joined_item_vii_int }
4468            { \int_use:N \l__enumext_columns_vii_int }
4469          \int_set:Nn \l__enumext_joined_item_vii_int
4470            {
4471              \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4472            }
4473        }
4474      \int_compare:nNnT
4475        { \l__enumext_joined_item_vii_int }
4476          >
4477        { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4478        {
4479          \msg_warning:nnee { enumext } { item-joined-columns }
4480            { \int_use:N \l__enumext_joined_item_vii_int }
4481            {
4482              \int_eval:n
4483                { \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1 }
4484            }
4485          \int_set:Nn \l__enumext_joined_item_vii_int
4486            {
4487              \l__enumext_columns_vii_int - \l__enumext_item_column_pos_vii_int + 1
4488            }
4489        }
4490      \int_compare:nNnTF { \l__enumext_joined_item_vii_int } > { 1 }
4491        {
4492          \int_set_eq:NN \l__enumext_joined_item_aux_vii_int \l__enumext_joined_item_vii_int
4493          \int_decr:N \l__enumext_joined_item_aux_vii_int
4494          \int_add:Nn \l__enumext_item_column_pos_vii_int { \l__enumext_joined_item_aux_vii_int }
4495          \int_gadd:Nn \g__enumext_item_count_all_vii_int { \l__enumext_joined_item_aux_vii_int }
4496          \dim_set:Nn \l__enumext_joined_width_vii_dim
4497            {
4498              \l__enumext_item_width_vii_dim * \l__enumext_joined_item_vii_int
4499              + (  \l__enumext_labelwidth_vii_dim + \l__enumext_labelsep_vii_dim
4500                 + \l__enumext_columns_sep_vii_dim
4501              )*\l__enumext_joined_item_aux_vii_int
4502            }
4503          \dim_set_eq:NN \itemwidth \l__enumext_joined_width_vii_dim
```

```
4504          }
4505          {
4506              \dim_set_eq:NN \l__enumext_joined_width_vii_dim \l__enumext_item_width_vii_dim
4507              \dim_set_eq:NN \itemwidth \l__enumext_item_width_vii_dim
4508          }
4509      }
```

Same implementation for the keyans* environment.

```
4510  \cs_new_protected:Npn \__enumext_starred_joined_item_viii:n #1
4511      {
4512          \int_set:Nn \l__enumext_joined_item_viii_int {#1}
4513          \int_compare:nNnT { \l__enumext_joined_item_viii_int } > { \l__enumext_columns_viii_int }
4514              {
4515                  \msg_warning:nnee { enumext } { item-joined }
4516                      { \int_use:N \l__enumext_joined_item_viii_int }
4517                      { \int_use:N \l__enumext_columns_viii_int }
4518                  \int_set:Nn \l__enumext_joined_item_viii_int
4519                      {
4520                          \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4521                      }
4522              }
4523          \int_compare:nNnT
4524              { \l__enumext_joined_item_viii_int }
4525                  >
4526              { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4527              {
4528                  \msg_warning:nnee { enumext } { item-joined-columns }
4529                      { \int_use:N \l__enumext_joined_item_viii_int }
4530                      {
4531                          \int_eval:n
4532                              { \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1 }
4533                      }
4534                  \int_set:Nn \l__enumext_joined_item_viii_int
4535                      {
4536                          \l__enumext_columns_viii_int - \l__enumext_item_column_pos_viii_int + 1
4537                      }
4538              }
4539          \int_compare:nNnTF { \l__enumext_joined_item_viii_int } > { 1 }
4540              {
4541                  \int_set_eq:NN \l__enumext_joined_item_aux_viii_int \l__enumext_joined_item_viii_int
4542                  \int_decr:N \l__enumext_joined_item_aux_viii_int
4543                  \int_add:Nn \l__enumext_item_column_pos_viii_int { \l__enumext_joined_item_aux_viii_int }
4544                  \int_gadd:Nn \g__enumext_item_count_all_viii_int { \l__enumext_joined_item_aux_viii_int }
4545                  \dim_set:Nn \l__enumext_joined_width_viii_dim
4546                      {
4547                          \l__enumext_item_width_viii_dim * \l__enumext_joined_item_viii_int
4548                          + ( \l__enumext_labelwidth_viii_dim + \l__enumext_labelsep_viii_dim
4549                              + \l__enumext_columns_sep_viii_dim
4550                          )*\l__enumext_joined_item_aux_viii_int
4551                      }
4552                  \dim_set_eq:NN \itemwidth \l__enumext_joined_width_viii_dim
4553              }
4554              {
4555                  \dim_set_eq:NN \l__enumext_joined_width_viii_dim \l__enumext_item_width_viii_dim
4556                  \dim_set_eq:NN \itemwidth \l__enumext_item_width_viii_dim
4557              }
4558      }
```

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

13.43.3 Functions for mini-env, mini-right and mini-right* keys

__enumext_start_mini_vii:
__enumext_stop_mini_vii:

The implementation of the mini-env key support is almost identical to the one used in the enumext and keyans environments, the difference is that the __enumext_mini_page environment on the *"right side"* is executed *"after"* closing the environment, so it is necessary to make a global copy of the variable \l__-enumext_minipage_right_vii_dim in the variable \g__enumext_minipage_right_vii_dim.

```
4559  \cs_new_protected:Nn \__enumext_start_mini_vii:
4560      {
4561          \dim_compare:nNnT { \l__enumext_minipage_right_vii_dim } > { \c_zero_dim }
4562              {
4563                  \dim_set:Nn \l__enumext_minipage_left_vii_dim
4564                      {
```

```
4565                \linewidth
4566                - \l__enumext_minipage_right_vii_dim
4567                - \l__enumext_minipage_hsep_vii_dim
4568              }
4569          \bool_set_true:N \l__enumext_minipage_active_vii_bool
4570          \dim_gset_eq:NN
4571            \g__enumext_minipage_right_vii_dim
4572            \l__enumext_minipage_right_vii_dim
4573          \__enumext_mini_addvspace_vii:
4574          \nointerlineskip\noindent
4575          \__enumext_mini_page{ \l__enumext_minipage_left_vii_dim }
4576        }
4577    }
```

The function __enumext_stop_mini_vii: closes the __enumext_mini_page environment on the *"left side"*, applies \hfill and set the variable \g__enumext_minipage_active_vii_bool to *"true"* which will be used in the function __enumext_after_env:nn to execute the minipage on the *"right side"*. At this point we will execute the __enumext_stop_list: and __enumext_stop_store_level_vii: functions stopping the list environment and the level saving mechanism for storage in *sequence* of the \anskey command and anskey* environment. This function is passed to the __enumext_after_list_vii: function in the second part of the enumext* environment definition (§13.44).

```
4578  \cs_new_protected:Nn \__enumext_stop_mini_vii:
4579    {
4580      \bool_if:NTF \l__enumext_minipage_active_vii_bool
4581        {
4582          \__enumext_stop_list:
4583          \__enumext_stop_store_level_vii:
4584          \IfDocumentMetadataTF { \tag_resume:n {enumext*} } { }
4585          \end__enumext_mini_page
4586          \hfill
4587          \bool_gset_true:N \g__enumext_minipage_active_vii_bool
4588        }
4589        {
4590          \__enumext_stop_list:
4591          \__enumext_stop_store_level_vii:
4592        }
4593    }
```

(*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.

```
4594  \__enumext_after_env:nn {enumext*}
4595    {
4596      \bool_if:NT \g__enumext_minipage_active_vii_bool
4597        {
4598          \__enumext_minipage:w [ t ] { \g__enumext_minipage_right_vii_dim }
4599            \legacy_if_gset_false:n { @minipage }
4600            \skip_vertical:N \c_zero_skip
4601            \par\addvspace { \g__enumext_minipage_right_skip }
4602            \bool_if:NF \g__enumext_minipage_center_vii_bool
4603              {
4604                \tl_put_left:Nn \g__enumext_miniright_code_vii_tl
4605                  {
4606                    \centering
4607                  }
4608              }
4609            \vbox_set_top:Nn \l__enumext_miniright_code_vii_box
4610              {
4611                \tl_use:N \g__enumext_miniright_code_vii_tl
4612              }
4613            \box_use_drop:N \l__enumext_miniright_code_vii_box
4614            \skip_vertical:N \c_zero_skip
4615          \__enumext_endminipage:
4616          \par\addvspace{ \g__enumext_minipage_after_skip }
4617        }
4618      \bool_gset_false:N \g__enumext_minipage_active_vii_bool
4619      \bool_gset_true:N \g__enumext_minipage_center_vii_bool
4620      \tl_gclear:N \g__enumext_miniright_code_vii_tl
4621      \dim_gzero:N \g__enumext_minipage_right_vii_dim
```

```
4622          \bool_gset_false:N \g__enumext_starred_bool
4623      }
```

__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.

```
4624  \cs_new_protected:Nn \__enumext_start_mini_viii:
4625    {
4626      \dim_compare:nNnT { \l__enumext_minipage_right_viii_dim } > { \c_zero_dim }
4627        {
4628          \dim_set:Nn \l__enumext_minipage_left_viii_dim
4629            {
4630              \linewidth
4631              - \l__enumext_minipage_right_viii_dim
4632              - \l__enumext_minipage_hsep_viii_dim
4633            }
4634          \bool_set_true:N \l__enumext_minipage_active_viii_bool
4635          \dim_gset_eq:NN
4636            \g__enumext_minipage_right_viii_dim
4637            \l__enumext_minipage_right_viii_dim
4638          \__enumext_mini_addvspace_viii:
4639          \nointerlineskip\noindent
4640          \__enumext_mini_page{ \l__enumext_minipage_left_viii_dim }
4641        }
4642    }
4643  \cs_new_protected:Nn \__enumext_stop_mini_viii:
4644    {
4645      \bool_if:NTF \l__enumext_minipage_active_viii_bool
4646        {
4647          \__enumext_stop_list:
4648          \IfDocumentMetadataTF { \tag_resume:n {keyans*} } { }
4649          \end__enumext_mini_page
4650          \hfill
4651          \bool_gset_true:N \g__enumext_minipage_active_viii_bool
4652        }
4653        {
4654          \__enumext_stop_list:
4655        }
4656    }
4657  \__enumext_after_env:nn {keyans*}
4658    {
4659      \bool_if:NT \g__enumext_minipage_active_viii_bool
4660        {
4661          \__enumext_mini_page{ \g__enumext_minipage_right_viii_dim }
4662            \par\addvspace { \g__enumext_minipage_right_skip }
4663            \bool_if:NF \g__enumext_minipage_center_viii_bool
4664              {
4665                \tl_put_left:Nn \g__enumext_miniright_code_viii_tl
4666                  {
4667                    \centering
4668                  }
4669              }
4670            \vbox_set_top:Nn \l__enumext_miniright_code_viii_box
4671              {
4672                \tl_use:N \g__enumext_miniright_code_viii_tl
4673              }
4674            \box_use_drop:N \l__enumext_miniright_code_viii_box
4675          \end__enumext_mini_page
4676          \par\addvspace{ \g__enumext_minipage_after_skip }
4677        }
4678      \bool_gset_false:N \g__enumext_minipage_active_viii_bool
4679      \bool_gset_true:N \g__enumext_minipage_center_viii_bool
4680      \tl_gclear:N \g__enumext_miniright_code_viii_tl
4681      \dim_gzero:N \g__enumext_minipage_right_viii_dim
4682    }
```

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

13.44   The environment enumext*

enumext* First we will generate the environment and we will give a temporary definition to __enumext_stop_-item_tmp_vii: equal to __enumext_first_item_tmp_vii: and next to \item equal to __enumext_-start_item_tmp_vii: which we will redefine later. Unlike the implementation used by the shortlst

package, we will not set the values of \rightskip and \@rightskip equal to \@flushglue whose value is 0.0pt plus 1.0 fil, in the tests I have performed this fails in some circumstances and different results are obtained when using pdfTEX and LuaTEX.

```
4683  \NewDocumentEnvironment{enumext*}{ o }
4684    {
4685      \__enumext_safe_exec_vii:
4686      \__enumext_parse_keys_vii:n {#1}
4687      \__enumext_before_list_vii:
4688      \__enumext_start_store_level_vii:
4689      \__enumext_start_list:nn { }
4690        {
4691          \__enumext_list_arg_two_vii:
4692          \__enumext_before_keys_exec_vii:
4693        }
4694      \IfDocumentMetadataTF { \tag_suspend:n {enumext*} } { }
4695      \__enumext_starred_columns_set_vii:
4696      \item[] \scan_stop:
4697      \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_first_item_tmp_vii:
4698      \cs_set_eq:NN \item \__enumext_start_item_tmp_vii:
4699      \ignorespaces
4700    }
4701    {
4702      \IfDocumentMetadataTF { \tag_struct_end:n {tag=text-unit} } { }
4703      \__enumext_stop_item_tmp_vii:
4704      \__enumext_remove_extra_parsep_vii:
4705      \__enumext_after_list_vii:
4706    }
```

(*End of definition for* enumext*. *This function is documented on page 5.*)

__enumext_safe_exec_vii: We will first call the function __enumext_is_not_nested: which sets \g__enumext_starred_bool to true if we are NOT nested within enumext, then call the function __enumext_internal_mini_page: to create the environment __enumext_mini_page, we will increment \l__enumext_level_h_int to restrict nesting of the environment, set \l__enumext_starred_bool to true and finally call the function __enumext_is_on_first_level: which sets \l__enumext_starred_first_bool to true if we are not nested, allowing the *"storage system"* to be used.

```
4707  \cs_new_protected:Nn \__enumext_safe_exec_vii:
4708    {
4709      \__enumext_is_not_nested:
4710      \__enumext_internal_mini_page:
4711      \int_incr:N \l__enumext_level_h_int
4712      \int_compare:nNnT { \l__enumext_level_h_int } > { 1 }
4713        {
4714          \msg_error:nn { enumext } { nested }
4715        }
4716      \int_compare:nNnT { \l__enumext_keyans_level_h_int } = { 1 }
4717        {
4718          \msg_error:nnn { enumext } { nested-horizontal } { keyans* }
4719        }
4720      \bool_set_true:N \l__enumext_starred_bool
4721      \bool_set_false:N \l__enumext_standar_bool
4722      \__enumext_is_on_first_level:
4723    }
```

(*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.

```
4724  \cs_new_protected:Npn \__enumext_parse_keys_vii:n #1
4725    {
4726      \tl_if_novalue:nF {#1}
4727        {
4728          \str_clear:N \l__enumext_series_str
4729          \keys_set:nn { enumext / enumext* } {#1}
4730          \__enumext_parse_series:n {#1}
4731          \__enumext_store_active_keys_vii:n {#1}
4732        }
4733    }
```

(*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*.

```
4734 \cs_new_protected:Nn \__enumext_before_list_vii:
4735   {
4736     \__enumext_vspace_above_vii:
4737     \__enumext_check_ans_active:
4738     \__enumext_before_args_exec_vii:
4739     \__enumext_start_mini_vii:
4740   }
```

(*End of definition for* __enumext_before_list_vii:.)

__enumext_after_list_vii: The function __enumext_after_list_vii: first calls the function __enumext_stop_mini_vii: which
internally calls __enumext_stop_list: and __enumext_stop_store_level_vii: (§13.43.3) used by
the keys mini-env, mini-right and mini-right*, then to the functions __enumext_after_stop_-
list_vii: used by the key after, __enumext_check_ans_key_hook: used by the key check-ans,
__enumext_vspace_below_vii: used by the keys below and below*. Finally set \l__enumext_-
starred_bool to false and call the __enumext_resume_save_counter: function used by the series,
resume and resume* keys.

```
4741 \cs_new_protected:Nn \__enumext_after_list_vii:
4742   {
4743     \__enumext_stop_mini_vii:
4744     \__enumext_after_stop_list_vii:
4745     \__enumext_check_ans_key_hook:
4746     \__enumext_vspace_below_vii:
4747     \bool_set_false:N \l__enumext_starred_bool
4748     \__enumext_resume_save_counter:
4749   }
```

(*End of definition for* __enumext_after_list_vii:.)

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

```
4750 \cs_new_protected:Nn \__enumext_start_store_level_vii:
4751   {
4752     \bool_if:NT \l__enumext_store_active_bool
4753       {
4754         \int_compare:nNnT { \l__enumext_level_int } > { 0 }
4755           {
4756             \__enumext_store_level_open_vii:
4757           }
4758       }
4759   }
4760 \cs_new_protected:Nn \__enumext_stop_store_level_vii:
4761   {
4762     \bool_if:NT \l__enumext_store_active_bool
4763       {
4764         \int_compare:nNnT { \l__enumext_level_int } > { 0 }
4765           {
4766             \__enumext_store_level_close_vii:
4767           }
4768       }
4769   }
```

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

13.44.1   The command \item in enumext*

__enumext_first_item_tmp_vii: The __enumext_first_item_tmp_vii: function will remove horizontal space equal to \labelwidth plus
\labelsep to the left of the *"first"* \item in the environment at the point of execution of this function, where
it is equal to the __enumext_stop_item_tmp_vii: function inside the environment body definition.

```
4770 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_vii:
4771   {
4772     \skip_horizontal:n
4773       {
4774         -\l__enumext_labelwidth_vii_dim - \l__enumext_labelsep_vii_dim
```

```
4775            }
4776        \ignorespaces
4777    }
```

(*End of definition for* __enumext_first_item_tmp_vii:.)

__enumext_start_item_tmp_vii:
__enumext_item_peek_args_vii:
__enumext_joined_item_vii:w
__enumext_standar_item_vii:w
__enumext_starred_item_vii:w
__enumext_starred_item_vii_aux_i:w
__enumext_starred_item_vii_aux_ii:w
__enumext_starred_item_vii_aux_iii:w

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

```
4778 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_vii:
4779    {
4780        \__enumext_stop_item_tmp_vii:
4781        \int_incr:N \l__enumext_item_column_pos_vii_int
4782        \int_gincr:N \g__enumext_item_count_all_vii_int
4783        \__enumext_item_peek_args_vii:
4784    }
```

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).

```
4785 \cs_new_protected:Nn \__enumext_item_peek_args_vii:
4786    {
4787        \peek_meaning:NTF (
4788            { \__enumext_joined_item_vii:w }
4789            { \__enumext_joined_item_vii:w (1) }
4790    }
```

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.

```
4791 \cs_new_protected:Npn \__enumext_joined_item_vii:w (#1)
4792    {
4793        \__enumext_starred_joined_item_vii:n {#1}
4794        \peek_meaning_remove:NTF *
4795            { \__enumext_starred_item_vii:w  }
4796            { \__enumext_standar_item_vii:w }
4797    }
```

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].

```
4798 \cs_new_protected:Npn \__enumext_standar_item_vii:w
4799    {
4800        \bool_set_false:N \l__enumext_item_starred_vii_bool
4801        \peek_meaning:NTF [
4802            {
4803                \bool_set_eq:NN \l__enumext_wrap_label_vii_bool \l__enumext_wrap_label_opt_vii_bool
4804                \__enumext_start_item_vii:w
4805            }
4806            {
4807                \bool_set_true:N \l__enumext_wrap_label_vii_bool
4808                \legacy_if_set_true:n { @noitemarg }
4809                \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ] \ignorespaces
4810            }
4811    }
```

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*⟩].

```
4812 \cs_new_protected:Npn \__enumext_starred_item_vii:w
4813    {
4814        \bool_set_true:N \l__enumext_item_starred_vii_bool
4815        \bool_set_true:N \l__enumext_wrap_label_vii_bool
4816        \peek_meaning:NTF [
4817            { \__enumext_starred_item_vii_aux_i:w }
```

```
4818                { \__enumext_starred_item_vii_aux_ii:w }
4819        }
4820    \cs_new_protected:Npn \__enumext_starred_item_vii_aux_i:w [#1]
4821        {
4822          \tl_gset:Nn \g__enumext_item_symbol_aux_vii_tl {#1}
4823          \__enumext_starred_item_vii_aux_ii:w
4824        }
4825    \cs_new_protected:Npn \__enumext_starred_item_vii_aux_ii:w
4826        {
4827          \peek_meaning:NTF [
4828            { \__enumext_starred_item_vii_aux_iii:w }
4829            {
4830              \dim_set_eq:NN \l__enumext_item_symbol_sep_vii_dim \l__enumext_labelsep_vii_dim
4831              \legacy_if_set_true:n { @noitemarg }
4832              \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ] \ignorespaces
4833            }
4834        }
4835    \cs_new_protected:Npn \__enumext_starred_item_vii_aux_iii:w [#1]
4836        {
4837          \dim_set:Nn \l__enumext_item_symbol_sep_vii_dim {#1}
4838          \legacy_if_set_true:n { @noitemarg }
4839          \__enumext_start_item_vii:w [ \l__enumext_label_vii_tl ] \ignorespaces
4840        }
```

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

__enumext_fake_make_label_vii:n   The __enumext_fake_make_label_vii:n function will be in charge of handling our definition of \item. First we increment the counter enumXvii for the enumerated items and activate support for the *check answers* mechanism, followed by support for \item*[⟨*symbol*⟩][⟨*offset*⟩] if present, then the wrap-label and wrap-label* keys which we execute using \makebox whose width will be given by the labelwidth key and position by the align key, inside the argument of this we will execute the font key together with the function defined by the wrap-label or wrap-label* keys. Finally we execute the labelsep key applying a \skip_horizontal:N and \ignorespaces.

🔷 For compatibility with *tagged* PDF and hyperref when an environment enumext is nested in enumext* and the key save-ans is not active need setting the \if@hyper@item switch to *"true"*. The explanation for this is given by the master Heiko Oberdiek on \refstepcounter{enumi} twice (or more) creates destination with the same identifier. This patch is only needed if you are running pdflatex and not if you are running lualatex

```
4841    \cs_new_protected_nopar:Npn \__enumext_fake_make_label_vii:n #1
4842        {
4843          \legacy_if:nT { @noitemarg }
4844            {
4845              \legacy_if_set_false:n { @noitemarg }
4846              \legacy_if:nT { @nmbrlist }
4847                {
4848                  \IfDocumentMetadataTF
4849                    {
4850                      \bool_if:NT \l__enumext_hyperref_bool
4851                        {
4852                          \legacy_if_set_true:n { @hyper@item }
4853                        }
4854                    } { }
4855                  \refstepcounter{enumXvii}
4856                  \bool_if:NT \l__enumext_check_answers_bool
4857                    {
4858                      \int_gincr:N \g__enumext_item_number_int
4859                      \bool_set_true:N \l__enumext_item_number_bool
4860                    }
4861                }
4862            }
4863          \bool_if:NT \l__enumext_item_starred_vii_bool
4864            {
4865              \tl_if_blank:VT \g__enumext_item_symbol_aux_vii_tl
4866                {
4867                  \tl_gset_eq:NN
4868                    \g__enumext_item_symbol_aux_vii_tl \l__enumext_item_symbol_vii_tl
4869                }
4870              \mode_leave_vertical:
4871              \skip_horizontal:n { -\l__enumext_item_symbol_sep_vii_dim }
4872              \hbox_overlap_left:n { \g__enumext_item_symbol_aux_vii_tl }
4873              \skip_horizontal:N \l__enumext_item_symbol_sep_vii_dim
```

```
4874          \tl_gclear:N \g__enumext_item_symbol_aux_vii_tl
4875        }
4876      \makebox[ \l__enumext_labelwidth_vii_dim ][ \l__enumext_align_label_vii_str ]
4877        {
4878          \tl_use:N \l__enumext_label_font_style_vii_tl
4879          \bool_if:NTF \l__enumext_wrap_label_vii_bool
4880            {
4881              \__enumext_wrapper_label_vii:n {#1}
4882            }
4883            { #1 }
4884        }
4885      \skip_horizontal:N \l__enumext_labelsep_vii_dim \ignorespaces
4886    }
```

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

13.44.2 Real definition of \item in enumext*

The functions __enumext_start_item_vii:w and __enumext_stop_item_vii: executing the true definition of \item inside the enumext* environment, unlike the implementation in shortlst we will NOT use an extra group and the plain form of the lrbox environment.

__enumext_start_item_vii:w
__enumext_stop_item_vii:

The first thing we will do is set the value of __enumext_stop_item_tmp_vii: equal to __enumext_-stop_item_vii: which we will define later, after that we will start capturing \item and *"item content"* in a *horizontal box* where the width will be \itemwidth plus \labelwidth plus \labelsep.

```
4887  \cs_new_protected_nopar:Npn \__enumext_start_item_vii:w [#1]
4888    {
4889      \cs_set_eq:NN \__enumext_stop_item_tmp_vii: \__enumext_stop_item_vii:
4890      \hbox_set_to_wd:Nnw \l__enumext_item_text_vii_box
4891        {
4892          \l__enumext_joined_width_vii_dim
4893          + \l__enumext_labelwidth_vii_dim
4894          + \l__enumext_labelsep_vii_dim
4895        }
```

Redefine the \footnote command.

```
4896        \__enumext_renew_footnote_starred:
```

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

```
4897        \__enumext_start_list_tag:n {enumext*}
4898        \__enumext_fake_make_label_vii:n {#1}
4899        \__enumext_stop_start_list_tag:
```

Finally we open the minipage environment, capture the *"item content"*, make \parindent take the value of the key listparindent and \parskip take the value of the key parsep, then execute the keys itemindent and first.

🎯 Here the use of \unskip and \skip_horizontal:n with the value of listparindent is necessary, otherwise an unwanted space is created when using \item[⟨*opt*⟩] and the value passed to the key itemindent is incremented.

```
4900        \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_vii_dim }
4901          \dim_set_eq:NN \parindent \l__enumext_listparindent_vii_dim
4902          \skip_set_eq:NN \parskip \l__enumext_parsep_vii_skip
4903          \__enumext_unskip_unkern:
4904          \__enumext_unskip_unkern:
4905          \skip_horizontal:n { -\l__enumext_listparindent_vii_dim } \ignorespaces
4906          \tl_use:N \l__enumext_fake_item_indent_vii_tl
4907          \tl_use:N \l__enumext_after_list_args_vii_tl
4908    }
```

The __enumext_stop_item_vii: function will finish the fetching \item and *"item content"* by closing the minipage environment, the *sockets* for *tagging* PDF and the *horizontal box*.

```
4909  \cs_new_protected_nopar:Nn \__enumext_stop_item_vii:
4910    {
4911        \__enumext_endminipage:
4912      \__enumext_stop_list_tag:n {enumext*}
4913      \hbox_set_end:
```

Here we will reduce the *warnings* a bit by setting the value of \hbadness to 10000, print \item and *"item content"* from the *horizontal box*,.

```
4914      \int_set:Nn \hbadness { 10000 }
4915      \box_use_drop:N \l__enumext_item_text_vii_box
```

Finally apply the *vertical space* between rows set by `itemsep` key passed to `\parsep` using `\par\noindent` and *horizontal space* between columns set by `columns-sep` key using `\skip_horizontal:N`.

```
4916      \int_compare:nNnTF
4917        { \l__enumext_item_column_pos_vii_int } = { \l__enumext_columns_vii_int }
4918        {
4919          \par\noindent
4920          \int_zero:N \l__enumext_item_column_pos_vii_int
4921        }
4922        {
4923          \skip_horizontal:N \l__enumext_columns_sep_vii_dim
4924        }
4925      }
```

(*End of definition for* `\__enumext_start_item_vii:w` *and* `\__enumext_stop_item_vii:`.)

`\__enumext_remove_extra_parsep_vii:` Remove the extra *vertical space* equal to `\parsep=\itemsep` when the total number of `\item` is divisible by the number of `\item` in the last row of the environment. Here the use of `\unskip` or `\removelastskip` fails and does not obtain the expected result, using `\vspace` is the option and in this case, we can use a simplified version since we are always in ⟨*vertical mode*⟩.

```
4926  \cs_new_protected:Nn \__enumext_remove_extra_parsep_vii:
4927    {
4928      \int_compare:nNnT
4929        {
4930          \int_mod:nn
4931            {  \g__enumext_item_count_all_vii_int } { \l__enumext_columns_vii_int }
4932        }
4933        =
4934        { 0 }
4935        {
4936          \para_end:
4937          \skip_vertical:n { -\l__enumext_itemsep_vii_skip }
4938          \skip_vertical:N \c_zero_skip
4939          \int_gzero:N \g__enumext_item_count_all_vii_int
4940        }
4941    }
```

(*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`.

```
4942  \__enumext_after_env:nn {enumext*}
4943    {
4944      \__enumext_execute_after_env:
4945    }
```

13.45   The environment keyans*

keyans* The implementation of `keyans*` environment is the similar as that used by the `enumext*` environment except for the `\__enumext_check_starred_cmd:n` function added in the second part.

```
4946  \NewDocumentEnvironment{keyans*}{ o }
4947    {
4948      \__enumext_safe_exec_viii:
4949      \__enumext_parse_keys_viii:n {#1}
4950      \__enumext_before_list_viii:
4951      \__enumext_start_list:nn { }
4952        {
4953          \__enumext_list_arg_two_viii:
4954          \__enumext_before_keys_exec_viii:
4955        }
4956      \IfDocumentMetadataTF { \tag_suspend:n {keyans*} } { }
4957      \__enumext_starred_columns_set_viii:
4958      \item[] \scan_stop:
4959      \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_first_item_tmp_viii:
4960      \cs_set_eq:NN \item \__enumext_start_item_tmp_viii:
4961      \ignorespaces
4962    }
4963    {
4964      \IfDocumentMetadataTF { \tag_struct_end:n {tag=text-unit} } { }
4965      \__enumext_stop_item_tmp_viii:
4966      \__enumext_remove_extra_parsep_viii:
4967      \__enumext_check_starred_cmd:n { item }
```

```
4968        \__enumext_after_list_viii:
4969      }
```

(*End of definition for* keyans*. *This function is documented on page* *15*.)

__enumext_safe_exec_viii: The __enumext_safe_exec_viii: function will first check if the save-ans key is active and only when this is true the environment will be available, it will increment the value of \l__enumext_keyans_level_h_int and return an error message when we are nesting the environment, then it will call the __enumext_-keyans_name_and_start: function in charge of saving the name of the environment and the line it is running on, then it will check if we are trying to nest keyans* in enumext* returning an error and we will set \l__enumext_starred_bool to true, finally we will check if we are within the appropriate level within the enumext environment.

```
4970  \cs_new_protected:Nn \__enumext_safe_exec_viii:
4971    {
4972      \bool_if:NF \l__enumext_store_active_bool
4973        {
4974          \msg_error:nnnn { enumext } { wrong-place }{ keyans* }{ save-ans }
4975        }
4976      \int_incr:N \l__enumext_keyans_level_h_int
4977      \int_compare:nNnT { \l__enumext_keyans_level_h_int } > { 1 }
4978        {
4979          \msg_error:nn { enumext } { nested }
4980        }
4981      \__enumext_keyans_name_and_start:
4982      \bool_if:NT \l__enumext_starred_bool
4983        {
4984          \msg_error:nnn { enumext } { nested-horizontal } { enumext* }
4985        }
4986      \bool_set_true:N \l__enumext_starred_bool
4987      % Set false for interfering with enumext nested in keyans* (yes, its possible and crayze)
4988      \bool_set_false:N \l__enumext_store_active_bool
4989      \int_compare:nNnT { \l__enumext_level_int } > { 1 }
4990        {
4991          \msg_error:nn { enumext } { keyans-wrong-level }
4992        }
4993    }
```

(*End of definition for* __enumext_safe_exec_viii:.)

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

```
4994  \cs_new_protected:Npn \__enumext_parse_keys_viii:n #1
4995    {
4996      \tl_if_novalue:nF {#1}
4997        {
4998          \keys_set:nn { enumext / keyans* } {#1}
4999        }
5000    }
```

(*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.

```
5001  \cs_new_protected:Nn \__enumext_before_list_viii:
5002    {
5003      \__enumext_vspace_above_viii:
5004      \__enumext_before_args_exec_viii:
5005      \__enumext_start_mini_viii:
5006    }
```

(*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.

```
5007  \cs_new_protected:Nn \__enumext_after_list_viii:
5008    {
5009      \__enumext_stop_mini_viii:
5010      \__enumext_after_stop_list_viii:
5011      \__enumext_vspace_below_viii:
5012    }
```

(*End of definition for* __enumext_after_list_viii:.)

13.45.1 The command `\item` in keyans*

The idea here is to make the `\item` command behave in the same way as in the `keyans` environment with the difference of the *optional argument* (⟨*number*⟩) which works in the same way as in the `enumext*` environment. In simple terms we want to store the ⟨*label*⟩ next to the [⟨*content*⟩] if it is present in the *sequence* and *prop list* defined by `save-ans` key for `\item*`, `\item*[`⟨*content*⟩`]`, `\item(`⟨*number*⟩`)*` and `\item(`⟨*number*⟩`)*[`⟨*content*⟩`]` commands.

`\__enumext_first_item_tmp_viii:`   The `\__enumext_first_item_tmp_viii:` function will remove horizontal space equal to `\labelwidth` plus `\labelsep` to the left of the *"first"* `\item` in the environment at the point of execution of this function, where it is equal to the `\__enumext_stop_item_tmp_viii:` function inside the environment body definition.

```
5013 \cs_new_protected_nopar:Nn \__enumext_first_item_tmp_viii:
5014   {
5015     \skip_horizontal:n
5016       {
5017         -\l__enumext_labelwidth_viii_dim - \l__enumext_labelsep_viii_dim
5018       }
5019     \ignorespaces
5020   }
```

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

`\__enumext_start_item_tmp_viii:`
`\__enumext_item_peek_args_viii:`
`\__enumext_joined_item_viii:w`
`\__enumext_standar_item_viii:w`

First we will call the function `\__enumext_stop_item_tmp_viii:` that we will redefine later, we will increment the value of `\l__enumext_item_column_pos_viii_int` that will count the item's by rows and the value of `\g__enumext_item_count_all_viii_int` that will count the total of item's in the environment. After that we will call the function `\__enumext_item_peek_args_viii:` that will handle the arguments passed to `\item`.

```
5021 \cs_new_protected_nopar:Nn \__enumext_start_item_tmp_viii:
5022   {
5023     \__enumext_stop_item_tmp_viii:
5024     \int_incr:N \l__enumext_item_column_pos_viii_int
5025     \int_gincr:N \g__enumext_item_count_all_viii_int
5026     \__enumext_item_peek_args_viii:
5027   }
```

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`).

```
5028 \cs_new_protected:Nn \__enumext_item_peek_args_viii:
5029   {
5030     \peek_meaning:NTF (
5031       { \__enumext_joined_item_viii:w }
5032       { \__enumext_joined_item_viii:w (1) }
5033   }
```

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`.

```
5034 \cs_new_protected:Npn \__enumext_joined_item_viii:w (#1)
5035   {
5036     \__enumext_starred_joined_item_viii:n {#1}
5037     \peek_meaning_remove:NTF *
5038       { \__enumext_starred_item_viii:w }
5039       { \__enumext_standar_item_viii:w }
5040   }
```

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 ]`.

```
5041 \cs_new_protected:Npn \__enumext_standar_item_viii:w
5042   {
5043     \bool_set_false:N \l__enumext_item_starred_viii_bool
5044     \bool_set_false:N \l__enumext_item_wrap_key_bool
5045     \peek_meaning:NTF [
5046       {
5047         \bool_set_eq:NN \l__enumext_wrap_label_viii_bool \l__enumext_wrap_label_opt_viii_bool
```

```
5048              \__enumext_start_item_viii:w
5049            }
5050          {
5051            \bool_set_true:N \l__enumext_wrap_label_viii_bool
5052            \legacy_if_set_true:n { @noitemarg }
5053            \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ] \ignorespaces
5054          }
5055      }
```

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

__enumext_starred_item_viii:w
__enumext_starred_item_viii_aux_i:w
__enumext_starred_item_viii_aux_ii:w
__enumext_keyans_starred_item_star:

The function __enumext_starred_item_viii:w together with the specified auxiliary functions aux_i:w and aux_ii:w execute \item* and \item*[⟨content⟩].

```
5056  \cs_new_protected:Npn \__enumext_starred_item_viii:w
5057      {
5058        \bool_set_true:N \l__enumext_item_starred_viii_bool
5059        \bool_set_true:N \l__enumext_item_wrap_key_bool
5060        \bool_set_true:N \l__enumext_wrap_label_viii_bool
5061        \peek_meaning:NTF [
5062          { \__enumext_starred_item_viii_aux_i:w }
5063          { \__enumext_starred_item_viii_aux_ii:w }
5064      }
```

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.

```
5065  \cs_new_protected:Npn \__enumext_starred_item_viii_aux_i:w [#1]
5066      {
5067        \tl_clear:N \l__enumext_store_current_label_tl
5068        \tl_if_novalue:nF { #1 }
5069          {
5070            \tl_if_empty:NF \l__enumext_store_keyans_item_opt_sep_tl
5071              {
5072                \tl_put_right:Ne \l__enumext_store_current_label_tl
5073                  {
5074                    \l__enumext_store_keyans_item_opt_sep_tl
5075                  }
5076                \tl_put_right:Ne \l__enumext_store_current_label_tl { #1 }
5077              }
5078            \tl_set:Ne \l__enumext_store_current_opt_arg_tl { #1 }
5079          }
5080        \__enumext_starred_item_viii_aux_ii:w
5081      }
5082  \cs_new_protected:Npn \__enumext_starred_item_viii_aux_ii:w
5083      {
5084        \legacy_if_set_true:n { @noitemarg }
5085        \__enumext_start_item_viii:w [ \l__enumext_label_viii_tl ] \ignorespaces
5086      }
```

The function __enumext_keyans_starred_item_star: will be in charge of storing the current ⟨label⟩ for \item* followed by the [⟨content⟩] for \item*[⟨content⟩] if present in the *sequence* and *prop list* set by the save-ans key. In this same function the keys show-ans, show-pos, mark-sep and save-ref are implemented.

```
5087  \cs_new_protected:Nn \__enumext_keyans_starred_item_star:
5088      {
5089        \tl_put_left:Ne \l__enumext_store_current_label_tl { \l__enumext_label_viii_tl }
5090        \__enumext_store_addto_prop:V \l__enumext_store_current_label_tl
5091        \__enumext_keyans_store_ref:
5092        \tl_put_left:Ne \l__enumext_store_current_label_tl { \item }
5093        \__enumext_keyans_addto_seq_link:
5094        \int_gincr:N \g__enumext_check_starred_cmd_int
5095        \dim_compare:nNnT { \l__enumext_mark_sym_sep_dim } = { \c_zero_dim }
5096          {
5097            \dim_set:Nn \l__enumext_mark_sym_sep_dim { \l__enumext_labelsep_viii_dim }
5098          }
5099        \bool_if:NT \l__enumext_show_answer_bool
5100          {
5101            \__enumext_print_keyans_box:NN
5102              \l__enumext_labelwidth_viii_dim \l__enumext_mark_sym_sep_dim
5103          }
```

```
5104        \bool_if:NT \l__enumext_show_position_bool
5105          {
5106            \tl_set:Ne \l__enumext_mark_answer_sym_tl
5107              {
5108                \group_begin:
5109                  \exp_not:N \normalfont
5110                  \exp_not:N \footnotesize [ \int_eval:n
5111                    {
5112                      \prop_count:c { g__enumext_ \l__enumext_store_name_tl _prop }
5113                    }
5114                  ]
5115                \group_end:
5116              }
5117            \__enumext_print_keyans_box:NN
5118              \l__enumext_labelwidth_viii_dim \l__enumext_mark_sym_sep_dim
5119          }
5120      }
```

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

`\__enumext_keyans_wraper_label_viii:n`
`\__enumext_fake_make_label_viii:n`

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

```
5121  \cs_new_protected:Npn \__enumext_keyans_wraper_label_viii:n #1
5122    {
5123      \bool_lazy_all:nT
5124        {
5125          { \bool_if_p:N \l__enumext_wrap_label_viii_bool     }
5126          { \bool_if_p:N \l__enumext_show_answer_bool          }
5127          { \bool_if_p:N \l__enumext_item_wrap_key_bool        }
5128          { \cs_if_exist_p:N \__enumext_keyans_wrapper_item:n }
5129        }
5130        {
5131          \cs_set_eq:NN \__enumext_wrapper_label_viii:n \__enumext_keyans_wrapper_item:n
5132        }
5133      \bool_if:NTF \l__enumext_wrap_label_viii_bool
5134        {
5135          \__enumext_wrapper_label_viii:n {#1}
5136        }
5137        { #1 }
5138    }
5139  \cs_new_protected_nopar:Npn \__enumext_fake_make_label_viii:n #1
5140    {
5141      \legacy_if:nT { @noitemarg }
5142        {
5143          \legacy_if_set_false:n { @noitemarg }
5144          \legacy_if:nT { @nmbrlist }
5145            {
5146              \refstepcounter{enumXviii}
5147            }
5148        }
5149      \bool_if:NT \l__enumext_item_starred_viii_bool
5150        {
5151          \__enumext_keyans_starred_item_star:
5152        }
5153      \makebox[ \l__enumext_labelwidth_viii_dim ][ \l__enumext_align_label_viii_str ]
5154        {
5155          \tl_use:N \l__enumext_label_font_style_viii_tl
5156          \__enumext_keyans_wraper_label_viii:n {#1}
5157        }
5158      \skip_horizontal:N \l__enumext_labelsep_viii_dim \ignorespaces
5159    }
```

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

13.45.2   Real definition of \item in keyans*

`\__enumext_start_item_viii:w`
`\__enumext_stop_item_viii:`

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

```
5160  \cs_new_protected_nopar:Npn \__enumext_start_item_viii:w [#1]
5161    {
5162      \cs_set_eq:NN \__enumext_stop_item_tmp_viii: \__enumext_stop_item_viii:
5163      \hbox_set_to_wd:Nnw \l__enumext_item_text_viii_box
5164        {
5165          \l__enumext_joined_width_viii_dim
```

```
5166          + \l__enumext_labelwidth_viii_dim
5167          + \l__enumext_labelsep_viii_dim
5168        }
5169      \__enumext_renew_footnote_starred:
5170      \__enumext_start_list_tag:n {keyans*}
5171      \__enumext_fake_make_label_viii:n {#1}
5172      \__enumext_stop_start_list_tag:
5173      \__enumext_minipage:w [ t ]{ \l__enumext_joined_width_viii_dim   }
5174        \dim_set_eq:NN \parindent \l__enumext_listparindent_viii_dim
5175        \skip_set_eq:NN \parskip \l__enumext_parsep_viii_skip
5176        \__enumext_unskip_unkern:
5177        \__enumext_unskip_unkern:
5178        \skip_horizontal:n { -\l__enumext_listparindent_viii_dim } \ignorespaces
5179        \tl_use:N \l__enumext_fake_item_indent_viii_tl
5180        \bool_if:NT \l__enumext_item_starred_viii_bool
5181          {
5182            \__enumext_keyans_show_item_opt:
5183          }
5184        \tl_use:N \l__enumext_after_list_args_viii_tl
5185    }
5186 \cs_new_protected_nopar:Nn \__enumext_stop_item_viii:
5187    {
5188      \__enumext_endminipage:
5189    \__enumext_stop_list_tag:n {keyans*}
5190    \hbox_set_end:
5191    \int_set:Nn \hbadness { 10000 }
5192    \box_use_drop:N \l__enumext_item_text_viii_box
5193    \int_compare:nNnTF
5194      { \l__enumext_item_column_pos_viii_int } = { \l__enumext_columns_viii_int }
5195      {
5196        \par\noindent
5197        \int_zero:N \l__enumext_item_column_pos_viii_int
5198      }
5199      {
5200        \skip_horizontal:N \l__enumext_columns_sep_viii_dim
5201      }
5202    }
```

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

`\__enumext_remove_extra_parsep_viii:`   The implementation at this is very similar to that of the enumext* environment.

```
5203 \cs_new_protected:Nn \__enumext_remove_extra_parsep_viii:
5204   {
5205     \int_compare:nNnT
5206       {
5207         \int_mod:nn
5208           { \g__enumext_item_count_all_viii_int }
5209           { \l__enumext_columns_viii_int }
5210       }
5211       =
5212       { 0 }
5213       {
5214         \para_end:
5215         \skip_vertical:n { -\l__enumext_itemsep_viii_skip }
5216         \skip_vertical:N \c_zero_skip
5217         \int_gzero:N \g__enumext_item_count_all_viii_int
5218       }
5219   }
```

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

13.46   The command \getkeyans

`\getkeyans`
`\__enumext_getkeyans_aux:n`
`\__enumext_getkeyans:nn`

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

```
5220 \NewDocumentCommand \getkeyans { m }
5221   {
5222     \exp_args:Ne \__enumext_getkeyans_aux:n
5223       { \tl_to_str:e { \text_expand:n {#1} } }
5224   }
```

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

```
5225 \cs_new_protected:Npn \__enumext_getkeyans_aux:n #1
5226   {
5227     \str_if_in:nnTF {#1} { : }
5228       {
5229         \use:e
5230           {
5231             \cs_set:Npn \exp_not:N \__enumext_tmp:w ##1 \c_colon_str ##2 \scan_stop:
5232               { {##1} {##2} }
5233           }
5234         \exp_after:wN \__enumext_getkeyans:nn \__enumext_tmp:w #1 \scan_stop:
5235       }
5236     { \msg_error:nnn { enumext } { missing-colon } {#1} }
5237   }
```

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*.

```
5238 \cs_new_protected:Npn \__enumext_getkeyans:nn #1 #2
5239   {
5240     \prop_if_exist:cTF { g__enumext_#1_prop }
5241       {
5242         \prop_item:cn { g__enumext_#1_prop }{#2}
5243       }
5244       {
5245         \msg_error:nnn { enumext } { undefined-storage-anskey } {#1}
5246       }
5247   }
```

(*End of definition for* `\getkeyans` *,* `\__enumext_getkeyans_aux:n` *, and* `\__enumext_getkeyans:nn` *. This function is documented on page 18.*)

13.47 The command `\printkeyans`

The `\printkeyans` command prints *"all stored content"* in the *sequence* defined by the `save-ans` key.
The first thing we will do is define a set of ⟨*filtered keys*⟩ with which we will control the options of the different nesting levels for the environment `enumext` and `enumext*` by storing their values in the list of tokens `\l__enumext_print_keyans_X_tl`.

The variable `\l__enumext_print_keyans_starred_tl` will have the default ⟨*keys*⟩ for `\printkeyans*` and will be set by `\setenumext[`⟨*print**⟩`]` and the variable `\l__enumext_print_keyans_vii_tl` will have the default keys for the environment `enumext*` nested within the *sequence* and will be set by `\setenumext[`⟨*print ,**⟩`]`, the rest of the variables will be for the environment `enumext` and will be set by `\setenumext[`⟨*print , level*⟩`]`.

```
5248 \keys_define:nn  { enumext / print }
5249   {
5250     print*  .code:n     = \keys_precompile:neN { enumext / enumext* }
5251                            { \__enumext_filter_save_key:n {#1} }
5252                            \l__enumext_print_keyans_starred_tl, % starred cmd
5253     print*  .initial:n = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5254                            rightmargin=0pt, listparindent=0pt, nosep, label=\arabic*.,
5255                            columns=2, first=\small, font=\small },
5256     print-1 .code:n     = \keys_precompile:neN { enumext / level-1 }
5257                            { \__enumext_filter_save_key:n {#1} }
5258                            \l__enumext_print_keyans_i_tl,
5259     print-1 .initial:n = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5260                            rightmargin=0pt, listparindent=0pt, nosep, label=\arabic*.,
5261                            columns=2, first=\small, font=\small },
5262     print-2 .code:n     = \keys_precompile:neN { enumext / level-2 }
5263                            { \__enumext_filter_save_key:n {#1} }
5264                            \l__enumext_print_keyans_ii_tl,
5265     print-2 .initial:n = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5266                            rightmargin=0pt, listparindent=0pt,  nosep, label=(\alph*),
5267                            first=\small, font=\small },
5268     print-3 .code:n     = \keys_precompile:neN { enumext / level-3 }
5269                            { \__enumext_filter_save_key:n {#1} }
5270                            \l__enumext_print_keyans_iii_tl,
5271     print-3 .initial:n = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5272                            rightmargin=0pt, listparindent=0pt, nosep, label=\roman*.,
5273                            first=\small, font=\small },
5274     print-4 .code:n     = \keys_precompile:neN { enumext / level-4 }
```

```
5275                                { \__enumext_filter_save_key:n {#1} }
5276                                \l__enumext_print_keyans_iv_tl,
5277        print-4 .initial:n  = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5278                                rightmargin=0pt, listparindent=0pt, nosep, label=\Alph*.,
5279                                first=\small, font=\small },
5280        print-* .code:n     = \keys_precompile:neN { enumext / enumext* }
5281                                { \__enumext_filter_save_key:n {#1} }
5282                                \l__enumext_print_keyans_vii_tl, % starred nested
5283        print-* .initial:n  = { labelwidth=0pt, labelsep=0.3333em, itemindent=0pt, list-offset=0pt,
5284                                rightmargin=0pt, listparindent=0pt, nosep, label=\arabic*.,
5285                                first=\small, font=\small },
5286      }
```

🫐 The reason for storing ⟨keys⟩ in token lists using \keys_precompile:neN is because the keys are set via \setenumext but are later executed by running the command \printkeyans and they are not handled directly by its *optional argument*, except those related to the *first* opening level.

\printkeyans
__enumext_printkeyans:nnn

Create a user command to print *"all stored content"* in *sequence* for \anskey, anskey*, \item* and \anspic*. Within a group we will run our *"precompiled keys"* and then call the internal function __enumext_-printkeyans:nnn.

```
5287  \NewDocumentCommand \printkeyans { s O{} m }
5288    {
5289      \group_begin:
5290        \tl_use:N \l__enumext_print_keyans_i_tl
5291        \tl_use:N \l__enumext_print_keyans_ii_tl
5292        \tl_use:N \l__enumext_print_keyans_iii_tl
5293        \tl_use:N \l__enumext_print_keyans_iv_tl
5294        \tl_use:N \l__enumext_print_keyans_vii_tl
5295        \__enumext_printkeyans:nnn { #1 } { #2 } { #3 }
5296      \group_end:
5297    }
```

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.

```
5298  \cs_new_protected:Npn \__enumext_printkeyans:nnn #1 #2 #3
5299    {
5300      \seq_if_exist:cTF { g__enumext_#3_seq }
5301        {
5302          \seq_if_empty:cF { g__enumext_#3_seq }
5303            {
```

If the *starred argument* '*' is present we will check that the environment enumext* is not saved in the *sequence*, then execute the variable \l__enumext_print_keyans_starred_tl that contains the default ⟨keys⟩ for the environment enumext*, we set \l__enumext_base_line_fix_bool and \l__enumext_print_keyans_-star_bool to true for *baseline correction*, open the enumext* environment passing the *optional argument* and map the *sequence*, then set \l__enumext_base_line_fix_bool and \l__enumext_print_keyans_-star_bool to false.

```
5304                \bool_if:nTF {#1}
5305                  {
5306                    \seq_if_in:cnTF { g__enumext_#3_seq } { \end{enumext*} }
5307                      {
5308                        \msg_error:nnnn { enumext } { print-starred } {#3} { enumext* }
5309                      }
5310                      {
5311                        \tl_use:N \l__enumext_print_keyans_starred_tl
5312                        \bool_set_true:N \l__enumext_base_line_fix_bool
5313                        \bool_set_true:N \l__enumext_print_keyans_star_bool
5314                        \begin{enumext*}[#2]
5315                          \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5316                        \end{enumext*}
5317                        \bool_set_false:N \l__enumext_base_line_fix_bool
5318                        \bool_set_false:N \l__enumext_print_keyans_star_bool
5319                      }
5320                  }
```

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

```
5321                  {
5322                    \begin{enumext}[#2]
5323                      \seq_map_inline:cn { g__enumext_#3_seq } { ##1 }
5324                    \end{enumext}
5325                  }
```

```
5326            }
5327          }
5328          {
5329            \msg_error:nnn { enumext } { undefined-storage-anskey } {#3}
5330          }
5331      }
```

(*End of definition for* \printkeyans *and* __enumext_printkeyans:nnn. *This function is documented on page 18.*)

13.48 The command \setenumext

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

__enumext_filter_first_level:n
__enumext_filter_first_level_key:n
__enumext_filter_first_level_pair:nn

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

```
5332  \cs_new:Npn \__enumext_filter_first_level:n #1
5333    {
5334      \use:e
5335        {
5336          \keyval_parse:NNn
5337            \__enumext_filter_first_level_key:n
5338            \__enumext_filter_first_level_pair:nn {#1}
5339        }
5340    }
```

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*.

```
5341  \cs_new:Npn \__enumext_filter_first_level_key:n #1
5342    {
5343      \str_case:nnF {#1}
5344        {
5345          { resume   } {}
5346          { resume*  } {}
5347        }
5348        { , { \exp_not:n {#1} } }
5349    }
```

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.

```
5350  \cs_new:Npn \__enumext_filter_first_level_pair:nn #1#2
5351    {
5352      \str_case:nnF {#1}
5353        {
5354          { series } {}
5355          { resume } {}
5356          { save-ans } {}
5357        }
5358        { , { \exp_not:n {#1} } = { \exp_not:n {#2} } }
5359    }
```

(*End of definition for* __enumext_filter_first_level:n, __enumext_filter_first_level_key:n, *and* __enumext_fil-ter_first_level_pair:nn.)

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

```
5360  \keys_define:nn { enumext / meta-families }
5361    {
5362      enumext-1 .code:n =
5363                {
5364                  \keys_set:ne { enumext / level-1 }
5365                    {
5366                      \__enumext_filter_first_level:n {#1}
5367                    }
5368                } ,
5369      enumext-2 .code:n = { \keys_set:nn { enumext / level-2 } {#1} } ,
5370      enumext-3 .code:n = { \keys_set:nn { enumext / level-3 } {#1} } ,
5371      enumext-4 .code:n = { \keys_set:nn { enumext / level-4 } {#1} } ,
5372      keyans    .code:n = { \keys_set:nn { enumext / keyans  } {#1} } ,
5373      enumext*  .code:n =
5374                {
5375                  \keys_set:ne { enumext / enumext* }
5376                    {
```

```
5377                        \__enumext_filter_first_level:n {#1}
5378                      }
5379                  } ,
5380        keyans*   .code:n = { \keys_set:nn { enumext / keyans* } {#1} } ,
5381        print*    .code:n = { \keys_set:nn { enumext / print    } { print*  = {#1} } } ,
5382        print-1   .code:n = { \keys_set:nn { enumext / print    } { print-1 = {#1} } } ,
5383        print-2   .code:n = { \keys_set:nn { enumext / print    } { print-2 = {#1} } } ,
5384        print-3   .code:n = { \keys_set:nn { enumext / print    } { print-3 = {#1} } } ,
5385        print-4   .code:n = { \keys_set:nn { enumext / print    } { print-4 = {#1} } } ,
5386        print-*   .code:n = { \keys_set:nn { enumext / print    } { print-* = {#1} } } ,
5387        unknown   .code:n = { \msg_error:nn { enumext } { unknown-key-family } } ,
5388      }
```

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

```
5389  \seq_const_from_clist:Nn \c__enumext_all_families_seq
5390    {
5391      enumext-1, enumext-2, enumext-3, enumext-4, keyans, enumext*,
5392      keyans*, print-1, print-2, print-3, print-4, print-*, print*,
5393    }
```

Now we define the user command \setenumext.

\setenumext

__enumext_set_parse:n

__enumext_set_error:nn

```
5394  \NewDocumentCommand \setenumext { O{enumext,1} +m }
5395    {
5396      \seq_clear:N \l__enumext_setkey_tmpa_seq
5397      \seq_set_from_clist:Nn \l__enumext_setkey_tmpb_seq {#1}
5398      \int_set:Nn \l__enumext_setkey_tmpa_int
5399        {
5400          \seq_count:N \l__enumext_setkey_tmpb_seq
5401        }
5402      \int_compare:nNnTF { \l__enumext_setkey_tmpa_int } > { 1 }
5403        {
5404          \seq_pop_left:NN \l__enumext_setkey_tmpb_seq \l__enumext_setkey_tmpa_tl
5405          \seq_map_function:NN \l__enumext_setkey_tmpb_seq \__enumext_set_parse:n
5406          \seq_set_map_e:NNn \l__enumext_setkey_tmpa_seq \l__enumext_setkey_tmpa_seq
5407            {
5408              \tl_use:N \l__enumext_setkey_tmpa_tl - ##1
5409            }
5410        }
5411        {
5412          \seq_put_right:Ne \l__enumext_setkey_tmpa_seq { \tl_trim_spaces:n {#1} }
5413        }
5414      \seq_if_empty:NTF \l__enumext_setkey_tmpa_seq
5415        { \seq_map_inline:Nn \c__enumext_all_families_seq }
5416        { \seq_map_inline:Nn \l__enumext_setkey_tmpa_seq }
5417        {
5418          \keys_set:nn { enumext / meta-families } { ##1 = {#2} }
5419        }
5420    }
```

Internal functions used by the \setenumext command.

```
5421  \cs_new_protected:Npn \__enumext_set_parse:n #1
5422    {
5423      \tl_set:Ne \l__enumext_setkey_tmpb_tl { \tl_trim_spaces:n {#1} }
5424      \clist_map_inline:nn { 0, 1, 2, 3, 4, * } % <- max level
5425        { \tl_remove_all:Nn \l__enumext_setkey_tmpb_tl {##1} }
5426      \tl_if_empty:NTF \l__enumext_setkey_tmpb_tl
5427        {
5428          \seq_put_right:Ne \l__enumext_setkey_tmpa_seq
5429            { \tl_trim_spaces:n {#1} }
5430        }
5431        { \__enumext_set_error:nn {#1} { } }
5432    }
5433  \cs_new_protected:Npn \__enumext_set_error:nn #1 #2
5434    { \msg_error:nnn { enumext } { invalid-key } {#1} {#2} }
```

(*End of definition for* \setenumext *,* __enumext_set_parse:n *, and* __enumext_set_error:nn*. This function is documented on page 6.*)

13.49   The command \setenumextmeta

The command \setenumextmeta will be responsible for adding new *"meta-keys"* for the enumext and enumext* environments. The implementation code was given by Jonathan P. Spratte (@Skillmon) answer in Add .meta key to existing keys (l3keys).

First we will create a prop list `\c__enumext_meta_paths_prop` to handle the *optional argument*.

```
5435 \prop_const_from_keyval:Nn \c__enumext_meta_paths_prop
5436   {
5437     {enumext,1} = level-1,
5438     {enumext,2} = level-2,
5439     {enumext,3} = level-3,
5440     {enumext,4} = level-4,
5441     {enumext*}  = enumext*
5442   }
```

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

```
5443 \NewDocumentCommand \setenumextmeta { s O{enumext,1} m +m }
5444   {
5445     \str_if_eq:eeTF { \tl_trim_spaces:n {#3} } { unknown }
5446       { \msg_error:nn { enumext } { prohibited-unknown } }
5447       {
5448         \bool_if:nTF {#1}
5449           {
5450             \int_step_inline:nn { 4 }
5451               { \__enumext_add_meta_key:nnn { enumext, ##1 } {#3} {#4} }
5452             \__enumext_add_meta_key:nnn { enumext* } {#3} {#4}
5453           }
5454           { \__enumext_add_meta_key:nnn {#2} {#3} {#4} }
5455       }
5456   }
```

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

```
5457 \cs_new_protected:Npn \__enumext_add_meta_key:nnn #1
5458   {
5459     \tl_set:Nn \l__enumext_meta_path_tl {#1}
5460     \tl_replace_all:Nnn \l__enumext_meta_path_tl { ~ } {}
5461     \prop_get:NVNTF
5462       \c__enumext_meta_paths_prop \l__enumext_meta_path_tl \l__enumext_meta_path_tl
5463       { \__enumext_def_meta_key:Vnn \l__enumext_meta_path_tl }
5464       {
5465         \msg_error:nnn { enumext } { unknown-set } {#1}
5466         \use_none:nn
5467       }
5468   }
5469 \cs_new_protected:Npn \__enumext_def_meta_key:nnn #1#2#3
5470   {
5471     \bool_lazy_or:nnTF
5472       { \keys_if_exist_p:nn { enumext / #1 } {#2} }
5473       { \keys_if_exist_p:nn { enumext / enumext* } {#2} }
5474       { \msg_error:nnn { enumext } { already-defined } {#2} }
5475       {
5476         \keys_define:nn { enumext / #1 }
5477           {
5478             #2 .meta:n = {#3},
5479             #2 .value_forbidden:n = true
5480           }
5481       }
5482   }
5483 \cs_generate_variant:Nn \__enumext_def_meta_key:nnn { V }
```

(*End of definition for* `\setenumextmeta` *and others. This function is documented on page 6.*)

13.50 The command `\foreachkeyans`

The command `\foreachkeyans` will execute a *loop* over the *prop list* and return its contents. The implementation code is adapted from the answer provided by Enrico Gregorio (@egreg) in Expand a `.cs` defined by key inside the function.

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

```
5484 \keys_define:nn { enumext / foreach }
5485   {
5486     before  .tl_set:N  = \l__enumext_foreach_before_tl,
5487     before  .value_required:n = true,
5488     after   .tl_set:N  = \l__enumext_foreach_after_tl,
5489     after   .value_required:n = true,
```

```
5490        start   .int_set:N = \l__enumext_foreach_start_int,
5491        start   .value_required:n = true,
5492        stop    .int_set:N = \l__enumext_foreach_stop_int,
5493        stop    .value_required:n = true,
5494        step    .int_set:N = \l__enumext_foreach_step_int,
5495        step    .value_required:n = true,
5496        wrapper .cs_set_protected:Np = \__enumext_foreach_wrapper:n #1,
5497        wrapper .value_required:n = true,
5498        sep     .tl_set:N  = \l__enumext_foreach_sep_tl,
5499        sep     .value_required:n = true,
5500        unknown .code:n    = { \__enumext_parse_foreach_keys:n {#1} }
5501      }
5502  \keys_precompile:nnN { enumext / foreach }
5503      {
5504        before={},after={},start=1,step=1,stop=0,wrapper=#1,sep={; }
5505      }
5506      \g__enumext_foreach_default_keys_tl
```

Functions for handling unknown ⟨keys⟩.

```
5507  \cs_new_protected:Npn \__enumext_parse_foreach_keys:nn #1#2
5508      {
5509        \tl_if_blank:nTF {#2}
5510          {
5511            \msg_error:nnn { enumext } { for-key-unknown } {#1}
5512          }
5513          {
5514            \msg_error:nnnn { enumext } { for-key-value-unknown } {#1} {#2}
5515          }
5516      }
5517  \cs_new_protected:Npn \__enumext_parse_foreach_keys:n #1
5518      {
5519        \exp_args:NV \__enumext_parse_foreach_keys:nn \l_keys_key_str {#1}
5520      }
```

We create the command.

```
5521  \NewDocumentCommand \foreachkeyans { +O{} m }
5522      {
5523        \__enumext_foreach_keyans:nn {#1} {#2}
5524      }
```

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

```
5525  \cs_new_protected:Npn \__enumext_foreach_keyans:nn #1 #2
5526      {
5527        \tl_use:N \g__enumext_foreach_default_keys_tl
5528        \keys_set:nn { enumext / foreach } {#1}
5529        \tl_set:Nn \l__enumext_foreach_name_prop_tl {#2}
5530        \prop_if_exist:cF { g__enumext_#2_prop }
5531          {
5532            \msg_error:nnn { enumext } { undefined-storage-anskey } {#2}
5533          }
5534        \int_compare:nNnT { \l__enumext_foreach_stop_int } = { 0 }
5535          {
5536            \int_set:Nn \l__enumext_foreach_stop_int
5537              { \prop_count:c { g__enumext_#2_prop } }
5538          }
5539        \seq_clear:N \l__enumext_foreach_print_seq
5540        \int_step_function:nnnN
5541          { \l__enumext_foreach_start_int }
5542          { \l__enumext_foreach_step_int }
5543          { \l__enumext_foreach_stop_int }
5544          \__enumext_foreach_add_body:n
5545        \seq_use:NV \l__enumext_foreach_print_seq \l__enumext_foreach_sep_tl
5546      }
5547  \cs_new_protected:Npn \__enumext_foreach_add_body:n #1
5548      {
5549        \seq_put_right:Ne \l__enumext_foreach_print_seq
5550          {
5551            \exp_not:V \l__enumext_foreach_before_tl
5552            \__enumext_foreach_wrapper:n
5553              {
5554                \prop_item:cn { g__enumext_ \l__enumext_foreach_name_prop_tl _prop }{#1}
```

```
5555              }
5556          \exp_not:V \l__enumext_foreach_after_tl
5557        }
5558    }
```

(*End of definition for* \foreachkeyans *and others. This function is documented on page* *18*.)

13.51   Messages

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

```
5559 \msg_new:nnn { enumext } { package-load }
5560    {
5561      The ~ '#1' ~ package ~ is ~ already ~ loaded.
5562    }
5563 \msg_new:nnn { enumext } { package-not-load }
5564    {
5565      The ~ '#1' ~ package ~ will ~ be ~ loaded ~ as ~ a ~ dependency.
5566    }
5567 \msg_new:nnn { enumext } { package-load-foot }
5568    {
5569      The ~ '#1' ~ package ~ is ~ loaded ~ with ~ the ~ option ~ '#2'.
5570    }
```

Message used in the creation of counters by enumext package.

```
5571 \msg_new:nnn { enumext } { counters }
5572    {
5573      The ~ counter ~ '#1' ~ is ~ already ~ defined ~ by ~ some ~ \\
5574      package ~ or ~ macro, ~ it ~ cannot ~ be ~ continued.
5575    }
```

Message used by align and mark-pos keys.

```
5576 \msg_new:nnn { enumext } { unknown-choice }
5577    {
5578      The ~ value ~ '#3' ~ for ~ '#1' ~ key ~ is ~ invalid ~ use ~ ('#2').
5579    }
```

Message used by reserved anskey* environment by enumext package.

```
5580 \msg_new:nnnn { enumext } { anskey-env-error }
5581    {
5582      The ~ '#1' ~ environment ~is ~ reserved ~ by ~\\
5583      'enumext' ~ package, ~ It~ is~ already~ defined.
5584    }
5585    {
5586      The ~ anskey* ~ environment ~ is ~ defined ~ internally ~
5587      for ~ the ~ 'save-ans' ~ key.\\
5588    }
```

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

```
5589 \msg_new:nnn { enumext } { store-prop }
5590    {
5591      * ~ Package ~ enumext: ~ Creating ~
5592        \c_backslash_str g__enumext_#1_prop ~ \msg_line_context:.
5593    }
5594 \msg_new:nnn { enumext } { store-seq }
5595    {
5596      * ~ Package ~ enumext: ~ Creating ~
5597      \c_backslash_str g__enumext_#1_seq ~ \msg_line_context:.
5598    }
5599 \msg_new:nnn { enumext } { store-int }
5600    {
5601      * ~ Package ~ enumext: ~ Creating ~
5602      \c_backslash_str g__enumext_resume_#1_int ~ \msg_line_context:.
5603    }
5604 \msg_new:nnn { enumext } { prop-seq-int-hook }
5605    {
5606      * ~ Package ~ enumext: ~ Elements ~ in ~
5607      \c_backslash_str g__enumext_#1_prop ~ = ~ #2.\\
5608      * ~ Package ~ enumext: ~ Elements ~ in ~
5609      \c_backslash_str g__enumext_#1_seq ~ = ~ #3.\\
5610      * ~ Package ~ enumext: ~ Value ~ off ~
5611      \c_backslash_str g__enumext_resume_#1_int ~ = ~ #4.
5612    }
5613 \msg_new:nnn { enumext } { item-answer-hook }
```

```
5614     {
5615       * ~ Package ~ enumext: ~ Value ~ off ~
5616       \c_backslash_str g__enumext_item_number_int ~ = ~ #1.\\
5617       * ~ Package ~ enumext: ~ Value ~ off ~
5618       \c_backslash_str g__enumext_item_anskey_int ~ = ~ #2.\\
5619       * ~ Package ~ enumext: ~ Difference ~ item_number_int ~ - ~ item_anskey_int ~ = ~ #3.
5620     }
```

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

```
5621  \msg_new:nnn { enumext } { invalid-key }
5622     {
5623       The ~ key ~ '#1' ~ is ~ not ~ know ~ the ~ level ~ #2.
5624     }
5625  \msg_new:nnn { enumext } { unknown-key-family }
5626     {
5627       Unknown~key~family~`\l_keys_key_str'~for~enumext.
5628     }
```

Messages used in length calculation.

```
5629  \msg_new:nnn { enumext } { width-negative }
5630     {
5631       Ignoring ~ negative ~ value ~ '#1=#2' ~ \msg_line_context:.\\
5632       The ~ key ~ '#1'~ accepts ~ values  ~ >= ~ 0pt.
5633     }
5634  \msg_new:nnn { enumext } { width-zero }
5635     {
5636       Invalid ~ '#1=#2' ~ \msg_line_context:.\\
5637       The ~ key ~ '#1'~ accepts ~ values  ~ > ~ 0pt.
5638     }
```

Messages used by show-length key in enumext.

```
5639  \msg_new:nnn { enumext } { list-lengths }
5640     {
5641       **** ~ Lengths ~ used ~ by ~ 'enumext' ~ level ~ '#2' ~ \msg_line_context:~\c_space_tl ****\\
5642       \__enumext_show_length:nnn { dim  } { labelsep      } {#1}
5643       \__enumext_show_length:nnn { dim  } { labelwidth    } {#1}
5644       \__enumext_show_length:nnn { dim  } { itemindent    } {#1}
5645       \__enumext_show_length:nnn { dim  } { leftmargin    } {#1}
5646       \__enumext_show_length:nnn { dim  } { rightmargin   } {#1}
5647       \__enumext_show_length:nnn { dim  } { listparindent } {#1}
5648       \__enumext_show_length:nnn { skip } { topsep     } {#1}
5649       \__enumext_show_length:nnn { skip } { parsep     } {#1}
5650       \__enumext_show_length:nnn { skip } { partopsep } {#1}
5651       \__enumext_show_length:nnn { skip } { itemsep    } {#1}
5652       **************************************************
5653     }
```

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

```
5654  \msg_new:nnn { enumext } { list-lengths-not-nested }
5655     {
5656       **** ~ Lengths ~ used ~ by ~ '#2' ~ environment ~ \msg_line_context:~\c_space_tl ****\\
5657       \__enumext_show_length:nnn { dim  } { labelsep      } {#1}
5658       \__enumext_show_length:nnn { dim  } { labelwidth    } {#1}
5659       \__enumext_show_length:nnn { dim  } { itemindent    } {#1}
5660       \__enumext_show_length:nnn { dim  } { leftmargin    } {#1}
5661       \__enumext_show_length:nnn { dim  } { rightmargin   } {#1}
5662       \__enumext_show_length:nnn { dim  } { listparindent } {#1}
5663       \__enumext_show_length:nnn { skip } { topsep     } {#1}
5664       \__enumext_show_length:nnn { skip } { parsep     } {#1}
5665       \__enumext_show_length:nnn { skip } { partopsep } {#1}
5666       \__enumext_show_length:nnn { skip } { itemsep    } {#1}
5667       **************************************************
5668     }
```

Messages used by ref key.

```
5669  \msg_new:nnn { enumext } { key-ref-empty }
5670     {
5671       Key ~ 'ref' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
5672     }
```

Messages used by save-ans key.

```
5673  \msg_new:nnn { enumext } { save-ans-empty }
5674     {
5675       Key ~ 'save-ans' ~ need ~ a ~ value ~ in ~ '#1'~ \msg_line_context:.
```

```
5676      }
5677  \msg_new:nnn { enumext } { save-ans-log }
5678      {
5679        * ~ Package ~ enumext: ~ Start ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5680      }
5681  \msg_new:nnn { enumext } { save-ans-log-hook }
5682      {
5683        * ~ Package ~ enumext: ~ Stop ~ #1\c_space_tl with ~ save-ans=#2 ~ \msg_line_context:.
5684      }
5685  \msg_new:nnn { enumext } { save-ans-hook }
5686      {
5687        Stop ~ storing ~ for ~ 'save-ans=#1' ~ \msg_line_context:.
5688      }
```

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

```
5689  \msg_new:nnn { enumext } { need-save-ans }
5690      {
5691        Key ~ '#1'~ works ~ only ~ with ~ the ~ 'save-ans' ~ key ~ in ~ '#2'~ \msg_line_context:.
5692      }
5693  \msg_new:nnn { enumext } { items-same-answer }
5694      {
5695        ***************************************\\
5696        * ~ Package ~ enumext: ~ Checking ~ answers ~ in ~ '#1' ~
5697        for ~ \c_left_brace_str #2 \c_right_brace_str\\
5698        * ~ started ~ #3 ~ and ~ close ~ \msg_line_context: : ~
5699        'OK', ~ all ~ items ~ with ~ answer.\\
5700        ***************************************
5701      }
5702  \msg_new:nnn { enumext } { item-greater-answer }
5703      {
5704        Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
5705        started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
5706        Items ~ > ~ Answers.
5707      }
5708  \msg_new:nnn { enumext } { item-less-answer }
5709      {
5710        Checking ~ answers ~ in ~ '#1' ~ for ~ \c_left_brace_str #2 \c_right_brace_str\\
5711        started ~ #3 ~ and ~ close ~ \msg_line_context: : ~'NOT ~ OK'\\
5712        Items ~ < ~ Answers.
5713      }
```

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

```
5714  \msg_new:nnn { enumext } { missing-starred }
5715      {
5716        Missing ~ '\c_backslash_str #1*' ~ #2.
5717      }
5718  \msg_new:nnn { enumext } { many-starred }
5719      {
5720        Many ~ '\c_backslash_str #1*' ~ #2.
5721      }
```

Messages used by \printkeyans* command.

```
5722  \msg_new:nnn { enumext } { print-starred }
5723      {
5724        \c_backslash_str printkeyans*:~ The ~ sequence ~ '#1' ~ already ~ contains ~
5725        #2 ~ environment ~  \msg_line_context:.
5726      }
```

Message for the nesting depth of the environment enumext.

```
5727  \msg_new:nnn { enumext } { list-too-deep }
5728      {
5729        Too ~ deep ~ nesting  ~ for ~ 'enumext' ~ \msg_line_context:.~ \\
5730        The ~ maximum  ~ level  ~ of  ~ nesting  ~ is ~ 4.
5731      }
```

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

```
5732  \msg_new:nnn { enumext } { anskey-unnumber-item }
5733      {
5734        Can't ~ store ~ with ~ a ~ unnumbered ~ \c_backslash_str item ~ \msg_line_context:.
5735      }
5736  \msg_new:nnn { enumext } { anskey-already-stored }
5737      {
5738        Content ~ already ~ stored ~ for ~ this ~ \c_backslash_str item ~ \msg_line_context:.
```

```
5739         }
5740  \msg_new:nnn { enumext } { anskey-empty-arg }
5741     {
5742        Can't ~ store ~ empty ~ content ~ \msg_line_context:.
5743     }
5744  \msg_new:nnn { enumext } { anskey-wrong-place }
5745     {
5746        Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
5747        '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5748     }
5749  \msg_new:nnn { enumext } { anskey-nested }
5750     {
5751        The ~ command ~ \c_backslash_str anskey~ can't ~ be ~ nested ~ \msg_line_context:.
5752     }
5753  \msg_new:nnn { enumext } { anskey-math-mode }
5754     {
5755        #1 ~ can't ~ work ~ in ~ math ~ mode ~ \msg_line_context:.
5756     }
5757  \msg_new:nnn { enumext } { anskey-env-wrong }
5758     {
5759        The ~ environment ~ anskey* ~ cannot ~ use ~ in ~ '#1' ~ \msg_line_context:.
5760     }
5761  \msg_new:nnn { enumext } { anspic-wrong-place }
5762     {
5763        Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
5764        '\c_backslash_str #1' ~ works ~ in ~ the ~ environment ~ '#2'.
5765     }
5766  \msg_new:nnn { enumext } { command-wrong-place }
5767     {
5768        Wrong ~ place ~ for ~ command ~ '\c_backslash_str #1' ~ \msg_line_context:.~ \\
5769        '\c_backslash_str #1' ~ works ~ outside ~ the ~ environment ~ '#2'.
5770     }
5771  \msg_new:nnnn { enumext } { anskey-env-key-unknown }
5772     {
5773        The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5774        'anskey*' ~ and ~ is ~ being ~ ignored.
5775     }
5776     {
5777        The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5778        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5779     }
5780  \msg_new:nnnn { enumext } { anskey-env-key-value-unknown }
5781     {
5782        The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5783        'anskey*' ~ and ~ is ~ being ~ ignored.
5784     }
5785     {
5786        The ~ environment ~ 'anskey*' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5787        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5788     }
5789  \msg_new:nnnn { enumext } { anskey-cmd-key-unknown }
5790     { The ~ key ~'#1'~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignored.}
5791     {
5792        The ~ command ~'\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5793        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5794     }
5795  \msg_new:nnnn { enumext } { anskey-cmd-key-value-unknown }
5796     { The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ '\c_backslash_str anskey' ~ and ~ is ~ being ~ ignor
5797     {
5798        The ~ command ~ '\c_backslash_str anskey' ~ does ~ not ~ have ~ a ~ key ~ called ~'#1'.\\
5799        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5800     }
```

Messages used by keyans, keyans* and keyanspic environment.

```
5801  \msg_new:nnn { enumext } { keyans-nested }
5802     {
5803        The ~ environment ~ 'keyans' ~ can't ~ be  ~ nested  ~ \msg_line_context:.
5804     }
5805  \msg_new:nnn { enumext } { keyans-wrong-level }
5806     {
5807        Wrong ~ level ~ position ~ for ~ 'keyans' ~ \msg_line_context:.~ \\
5808        The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
```

```
5809      }
5810  \msg_new:nnn { enumext } { wrong-place }
5811      {
5812        Wrong ~ place ~ for ~ '#1' ~ environment ~\msg_line_context:.~ \\
5813        '#1' ~ is ~ only ~ found ~ with ~ '#2' ~  in  ~  'enumext.
5814      }
5815  \msg_new:nnn { enumext } { keyanspic-nested }
5816      {
5817        The ~ environment ~ 'keyanspic' ~ can't ~ be  ~ nested~ \msg_line_context:.~.
5818      }
5819  \msg_new:nnn { enumext } { keyanspic-wrong-level }
5820      {
5821        Wrong ~ level ~ position ~ for ~ 'keyanspic' ~ \msg_line_context:.~ \\
5822        The ~ environment ~ 'keyans' ~ can ~ only ~ be ~ in ~ the ~ first ~ level.
5823      }
5824  \msg_new:nnn { enumext } { keyanspic-item-cmd }
5825      {
5826        Can't ~ use  ~ \c_backslash_str item ~ in ~ keyanspic ~ \msg_line_context:.
5827      }
5828  \msg_new:nnnn { enumext } { keyans-unknown-key }
5829      {
5830        The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5831        '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5832      }
5833      {
5834        The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5835        ~ have ~ a ~ key ~ called ~'#1'.\\
5836        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5837      }
5838  \msg_new:nnnn { enumext } { keyans-unknown-key-value }
5839      {
5840        The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5841        '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5842      }
5843      {
5844        The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5845        ~ have ~ a ~ key ~ called ~'#1'.\\
5846        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5847      }
```

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

```
5848  \msg_new:nnnn { enumext } { starred-unknown-key }
5849      {
5850        The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment~
5851        '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5852      }
5853      {
5854        The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5855        ~ have ~ a ~ key ~ called ~'#1'.\\
5856        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5857      }
5858  \msg_new:nnnn { enumext } { starred-unknown-key-value }
5859      {
5860        The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~
5861        '\l__enumext_envir_name_tl' ~ and ~ is ~ being ~ ignored.
5862      }
5863      {
5864        The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5865        ~ have ~ a ~ key ~ called ~'#1'.\\
5866        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5867      }
```

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

```
5868  \msg_new:nnnn { enumext } { standar-unknown-key }
5869      {
5870        The ~ key ~ '#1' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_tl
5871        ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5872      }
5873      {
5874        The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5875        ~ have ~ a ~ key ~ called ~'#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5876        Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
```

```
5877     }
5878  \msg_new:nnnn { enumext } { standar-unknown-key-value }
5879     {
5880       The ~ key ~ '#1=#2' ~ is ~ unknown ~ by ~ environment ~ '\l__enumext_envir_name_tl' \c_space_
5881       ~ on ~ level ~ \int_use:N \l__enumext_level_int \c_space_tl and ~ is ~ being ~ ignored.
5882     }
5883     {
5884       The ~ environment ~ '\l__enumext_envir_name_tl' ~ does ~ not
5885       ~ have ~ a ~ key ~ called ~'#1' ~ on ~ level ~ \int_use:N \l__enumext_level_int.\\
5886       Check ~ that ~ you ~ have ~ spelled ~ the ~ key ~ name ~ correctly.
5887     }
```

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

```
5888  \msg_new:nnnn { enumext } { for-key-unknown }
5889     { The~key~'#1'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored.}
5890     {
5891       The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5892       Check~that~you~have~spelled~the~key~name~correctly.
5893     }
5894  \msg_new:nnnn { enumext } { for-key-value-unknown }
5895     { The~key~'#1=#2'~is~unknown~by~'\c_backslash_str foreachkeyans'~and~is~being~ignored. }
5896     {
5897       The~command~'\c_backslash_str foreachkeyans'~does~not~have~a~key~called~'#1'.\\
5898       Check~that~you~have~spelled~the~key~name~correctly.
5899     }
```

Messages used by \getkeyans command.

```
5900  \msg_new:nnn { enumext } { undefined-storage-anskey }
5901     {
5902       Storage ~ named ~ '#1' ~ is ~ not ~ defined ~ \msg_line_context:.
5903     }
```

Messages used by \miniright command.

```
5904  \msg_new:nnn { enumext } { missing-miniright }
5905     {
5906       Missing ~ '\c_backslash_str miniright' ~ in ~ \msg_line_context:.\\
5907       The ~ key ~ 'mini-env' ~ need ~ '\c_backslash_str miniright'.
5908     }
5909  \msg_new:nnn { enumext } { wrong-miniright-place }
5910     {
5911       Wrong ~ place ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
5912       Works ~ in ~ 'enumext' ~ and ~ 'keyans' ~ with ~ key ~ 'mini-env'.
5913     }
5914  \msg_new:nnn { enumext } { wrong-miniright-use }
5915     {
5916       Wrong ~ use ~ for ~ '\c_backslash_str miniright' ~ \msg_line_context:.~ \\
5917       '\c_backslash_str miniright' ~ need ~ a ~ key ~ 'mini-env'.
5918     }
5919  \msg_new:nnn { enumext } { wrong-miniright-starred }
5920     {
5921       Can't ~ use  ~ \c_backslash_str miniright ~ in ~ starred ~ environments ~ \msg_line_context:.
5922     }
5923  \msg_new:nnn { enumext } { many-miniright-used }
5924     {
5925       Can't ~ use  ~ \c_backslash_str miniright ~ more ~ than ~ once ~  \msg_line_context:.
5926     }
```

Messages used by \setenumextmeta command.

```
5927  \msg_new:nnn { enumext } { unknown-set }
5928     {
5929       Argument ~ [#1] ~ is ~ unknown ~ by ~  \c_backslash_str setenumextmeta ~ \msg_line_context:.
5930     }
5931  \msg_new:nnn { enumext } { already-defined }
5932     {
5933       The ~ key ~ '#1' ~ is ~ already ~ defined ~ \msg_line_context:.
5934     }
5935  \msg_new:nnn { enumext } { prohibited-unknown }
5936     {
5937       The ~ name ~ 'unknown' ~ can't ~ be ~ chosen~ for ~ a ~ meta ~ key ~ \msg_line_context:.
5938     }
```

Messages used by enumext* and keyans* environments.

```
5939  \msg_new:nnn { enumext } { nested }
```

```
5940      {
5941        The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ \msg_line_con
5942      }
5943  \msg_new:nnn { enumext } { nested-horizontal }
5944      {
5945        The ~ environment ~ \l__enumext_envir_name_tl \c_space_tl can't ~ be ~ nested ~ in ~ '#1' ~ '
5946      }
5947  \msg_new:nnn { enumext } { item-joined }
5948      {
5949        Items ~ joined ~ (#1) ~ > ~ #2  ~ columns ~\msg_line_context:.
5950      }
5951  \msg_new:nnn { enumext } { item-joined-columns }
5952      {
5953        Not ~ space ~ to ~ join ~ items ~ (#1) ~ > ~ #2 ~\msg_line_context:.
5954      }
```

13.52 Finish package

Finish package implementation.

```
5955  \file_input_stop:
5956  ⟨/package⟩
```

14　Index of Implementation

The italic numbers denote the pages where the corresponding entry is described, the numbers underlined and all others indicate the line on which they are implemented in the package code.